

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 ІІ-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 ІІ-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

/п	Назва етапів виконання курсового проекту	Строк виконання етапів проекту	Примітка
1	Аналіз предметного середовища	12.11.2022	
2	Побудова ER-моделі	21.11.2023	
3	Побудова реляційної схеми з ER-моделі	01.12.2022	
4	Створення бази даних, у форматі обраної системи управління базою даних	01.12.2022	
5	Створення користувачів бази даних	08.01.2023	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	20.12.2022	
7	Створення мовою SQL запитів	07.01.2023	
8	Оптимізація роботи запитів	08.01.2023	
9	Оформлення пояснювальної записки	08.01.2023	
10	Захист курсової роботи	11.01.2022	

Студент

(підпис) (прізвище та ініціали)

Керівник роботи

(підпис) (прізвище та ініціали)

ЗМІСТ

Вступ.....	6
1. Опис предметного середовища.....	7
2. Постановка завдання	8
3. Проектування бази даних.....	9
3.1. Побудова ER-моделі.....	9
3.1.1. Бізнес-правила	9
3.1.2. Вибір сутностей.....	9
3.1.3. Набори атрибутів сутностей	10
3.2. Даталогічна модель бази даних	14
4. Реалізація бази даних.....	16
4.1. Створення бази даних у форматі СУБД PostgreSQL	16
4.1.1. Створення таблиць	16
4.1.2. Створення зав'язків	17
4.1.3. Створення обмежень	19
4.2. Імпортування даних у таблиці	19
5. Робота з базою даних.....	20
5.1. Створення користувачів	20
5.1.1. Абітурієнт	20
5.1.2. Член приймальної комісії.....	20
5.1.3. Декан	21
5.2. SQL запити.....	22
5.2.1. Тригери, що будуть змінюватись користувачами.....	22
5.2.2. Процедури та функцій для взаємодії користувачів з базою даних	24

	5
5.2.3. Представлення	30
5.2.4. DML-запити	31
5.2.6. Індокси.....	44
Висновки.....	46
Список використаної літератури.....	47
Додаток А. Тексти INSERT запитів	48

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 сформулювати наказ про зарахування.

2. 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

3. ПРОЕКТУВАННЯ БАЗИ ДАНИХ

3.1. Побудова ER-моделі

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

3.1.1. 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;

3.1.2. Вибір сутностей

- Зарахований абітурієнт
- Спеціальність
- Заява
- Пріоритет по кафедрам
- Кафедра
- Факультет
- Документ
- Абітурієнт
- Результат ЗНО
- Предмет ЗНО

3.1.3. 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.

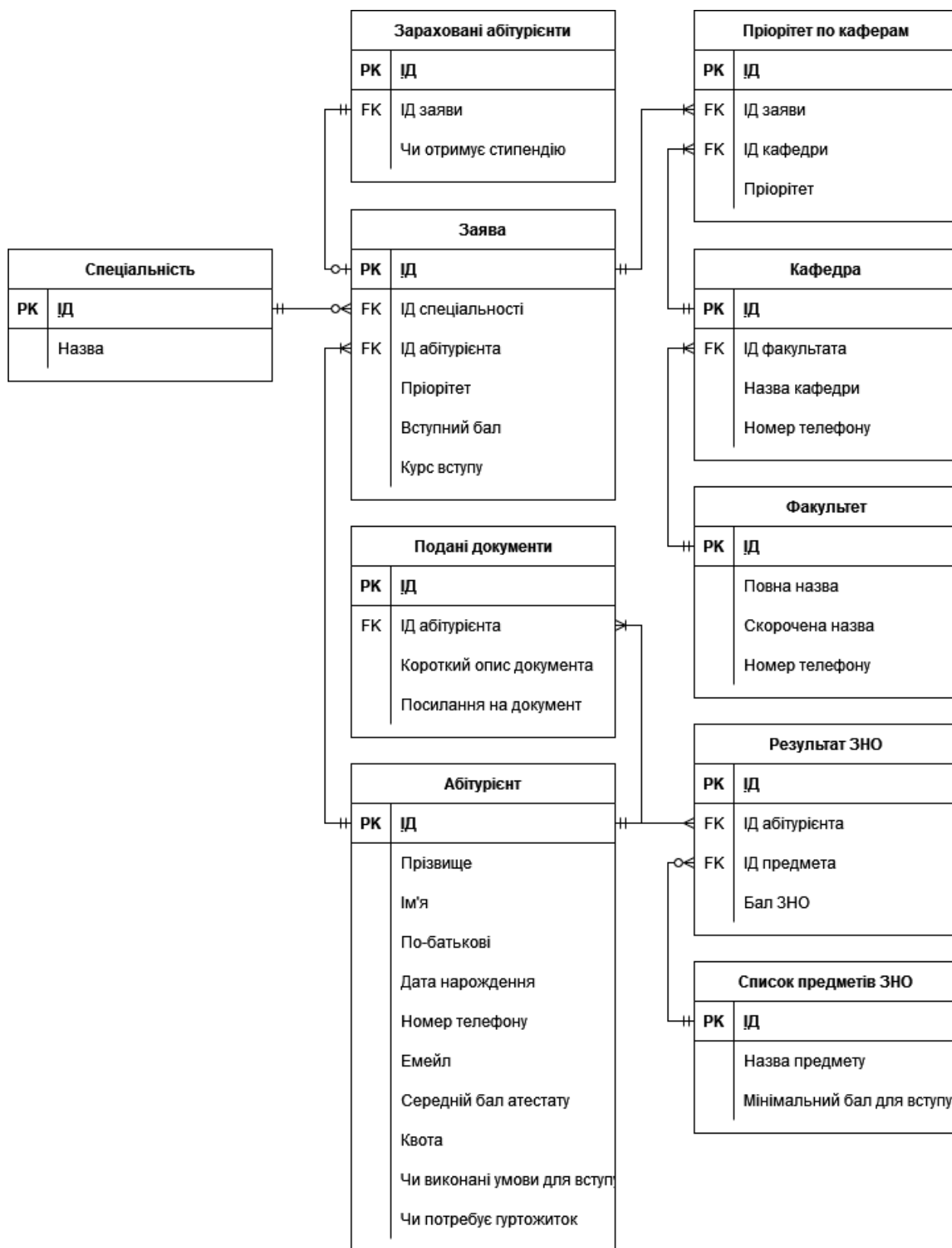


Рисунок 3.1 – ER-діаграма

3.2. Даталогічна модель бази даних

3.2.1. Побудова необхідних відношень та визначення первинних і зовнішніх ключів

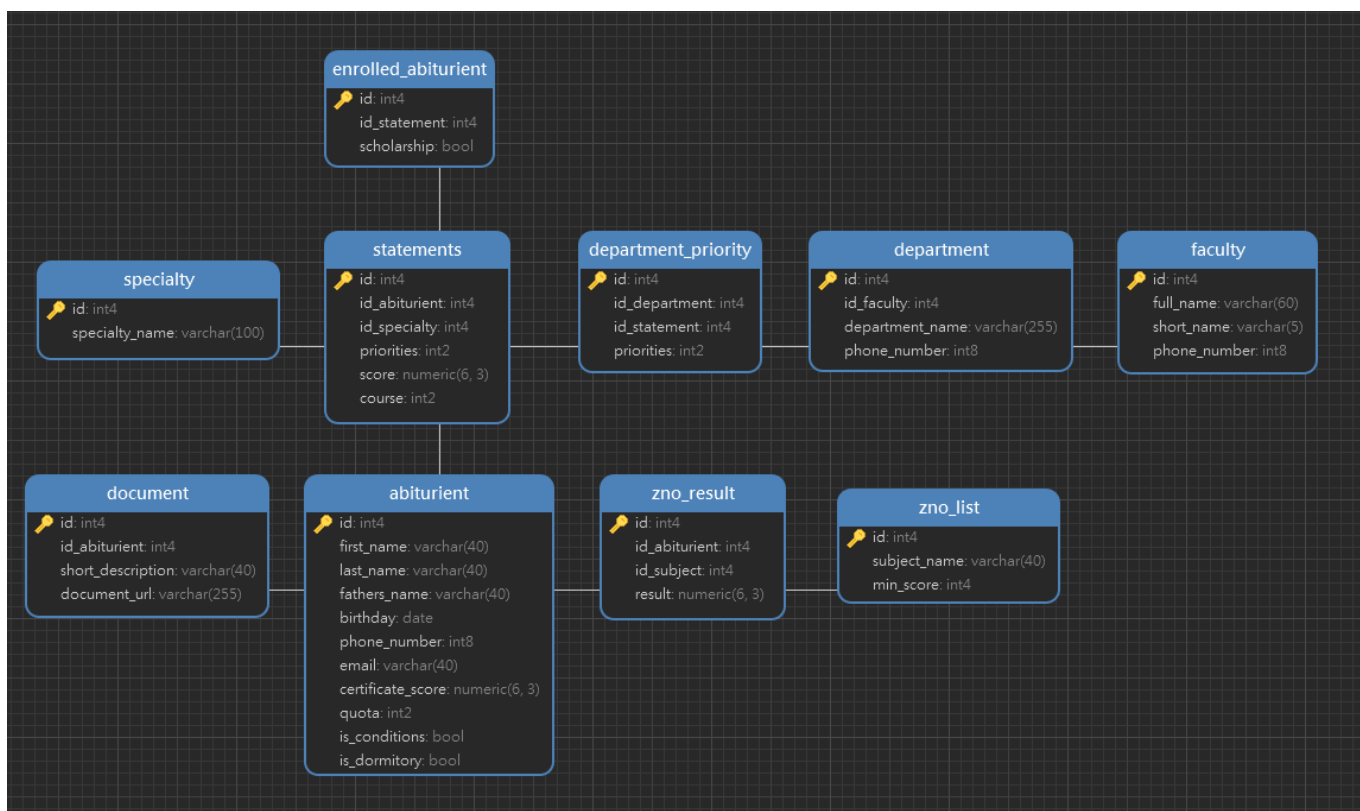


Рисунок 3.2 – Даталогічна модель бази даних

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 значить `ON DELETE RESTRICT`;

- Обов'язкові атрибути таблиць мають обмеження NOT NULL, для запобігання помилок при роботі з даними.

4. РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

4.1. Створення бази даних у форматі СУБД PostgreSQL

4.1.1. 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
);

```

4.1.2. 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);
```

4.1.3. 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);

```

4.2. 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.

5. РОБОТА З БАЗОЮ ДАНИХ

5.1. Створення користувачів

5.1.1. Абитурієнт

Текст скрипта:

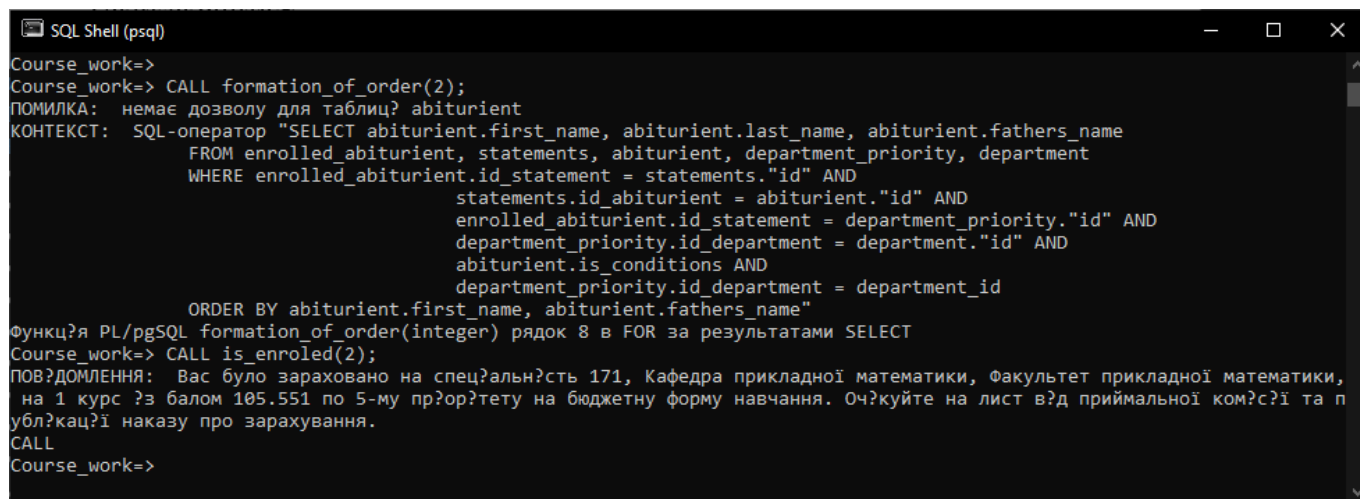
```
CREATE ROLE abiturient NOLOGIN;

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_enrolled(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:



```
SQL Shell (psql)
Course_work=>
Course_work=> CALL formation_of_order(2);
ПОМИЛКА: немає дозволу для таблиці abiturient
КОНТЕКСТ:  SQL-оператор "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"
Функція PL/pgSQL formation_of_order(integer) рядок 8 в FOR за результатами SELECT
Course_work=> CALL is_enrolled(2);
ПОВІДОМЛЕННЯ:  Вас було зараховано на спеціальність 171, Кафедра прикладної математики, Факультет прикладної математики,
на 1 курс з балом 105.551 по 5-му прорітету на бюджетну форму навчання. Очкуйте на лист в'їд приймальної комісії та п
ублікації наказу про зарахування.
CALL
Course_work=>
```

Рисунок 5.1 – Результат виконання скрипта

5.1.2. Член приймальної комісії

Текст скрипта:

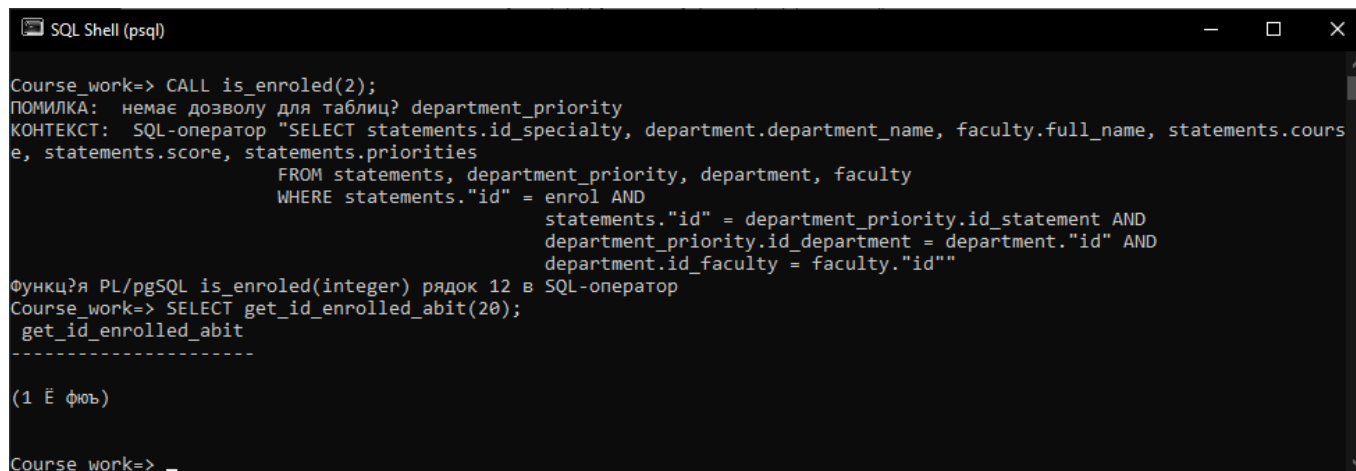
```
CREATE ROLE selection_committee NOLOGIN;

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:



```
SQL Shell (psql)
Course_work=> CALL is_enrolled(2);
ПОМИЛКА: немає дозволу для таблиц? department_priority
КОНТЕКСТ: SQL-оператор "SELECT statements.id_specialty, department.department_name, faculty.full_name, statements.cours
e, 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""
Функція PL/pgSQL is_enrolled(integer) рядок 12 в SQL-оператор
Course_work=> SELECT get_id_enrolled_abit(20);
get_id_enrolled_abit
-----
(1 Є фюъ)
Course_work=> _
```

Рисунок 5.2 – Результат виконання скрипта

5.1.3. Декан

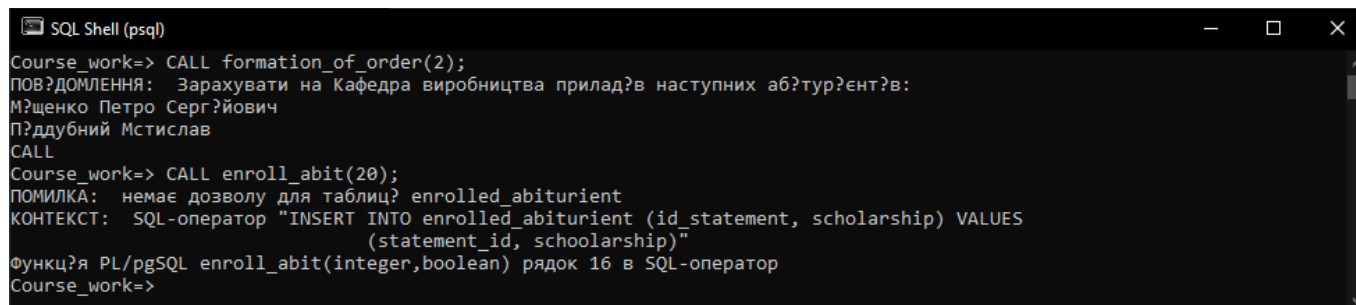
Текст скрипта:

```
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:



```
SQL Shell (psql)
Course_work=> CALL formation_of_order(2);
ПОВ?ДОМЛЕННЯ: Зарахувати на Кафедра виробництва прилад?в наступних аб?тур?ент?в:
М?щенко Петро Серг?йович
П?ддубний Мстислав
CALL
Course_work=> CALL enroll_abit(20);
ПОМИЛКА: немає дозволу для таблиц? enrolled_abiturient
КОНТЕКСТ: SQL-оператор "INSERT INTO enrolled_abiturient (id_statement, scholarship) VALUES
(statement_id, scholarship)"
Функція PL/pgSQL enroll_abit(integer,boolean) рядок 16 в SQL-оператор
Course_work=> _
```

Рисунок 5.3 – Результат виконання скрипта

5.2.SQL запити

5.2.1. 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:

```

Message
CALL enroll_abit(8)
> ПОВІДОМЛЕННЯ:
        Новий абітурієнт
        ПІБ: Балицька Ірина Володимирівна
        Пошта: ol070506@gmail.com

```

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:

```

> ПОВІДОМЛЕННЯ: SEND LETTER;
send_to=ol070506@gmail.com;
text="Шановний(на) Балицька Ірина Володимирівна\nВас зараховано на 1, Кафедра економічної кібернетики,
ФММ із вступним балом 117.660. Вам потрібно буде з'явитися в деканаті протягом тижня з дня надходження
цього листа для підписання документів.\n\nЗ повагою, Деканат";

> ПОВІДОМЛЕННЯ: Задану заяву було успішно зараховано
> OK
> Time: 0,015s

```

Рисунок 5.5 – Результат виконання скрипта

5.2.2. Процедури та функцій для взаємодії користувачів з базою даних

5.2.2.1. 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"

```



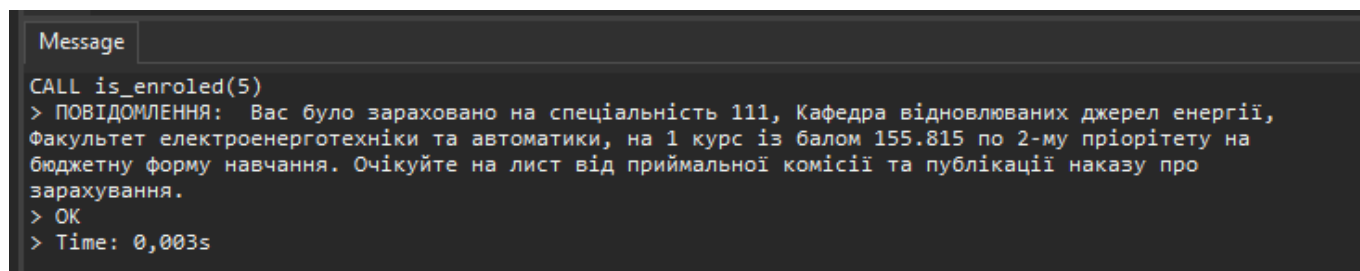
```

        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 пріорбтету 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:



```

Message
CALL is_enroled(5)
> ПОВІДОМЛЕННЯ: Вас було зараховано на спеціальність 111, Кафедра відновлюваних джерел енергії,
Факультет електроенерготехніки та автоматики, на 1 курс із балом 155.815 по 2-му пріорітету на
бюджетну форму навчання. Очікуйте на лист від приймальної комісії та публікації наказу про
зарахування.
> OK
> Time: 0,003s

```

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);
        scholarship BOOLEAN;
    BEGIN
        IF (enrol IS NOT NULL) THEN
            SELECT enrolled_abiturient.scholarship
            FROM enrolled_abiturient
            WHERE enrolled_abiturient.id_statement = enrol
            INTO scholarship;

            IF (schollarship) THEN
                RAISE NOTICE 'Ви отримуватимете стипендію';
            END IF;
        END IF;
    END;
$$

```

```

        ELSE
            RAISE NOTICE 'Ви НЕ отримуватимете стипендію';
        END IF;
    ELSE
        RAISE NOTICE 'Вас ще не було зараховано';
    END IF;
END;
$$ LANGUAGE plpgsql;

```

Результат:

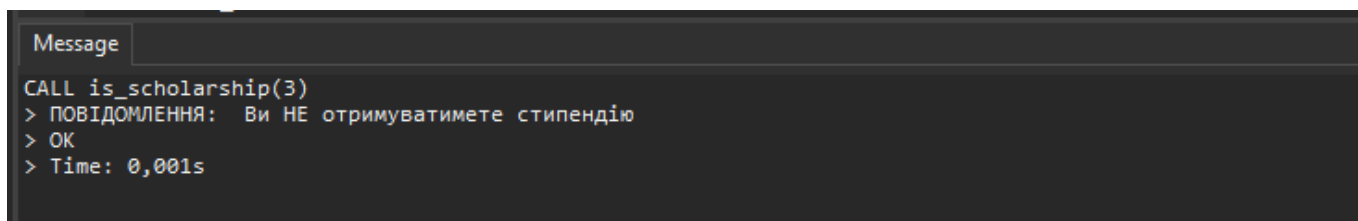


Рисунок 5.7 – Результат виконання скрипта

- 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:

specialty	prioritet	score	course
121	1	196,300	1
123	2	143,250	1
111	3	119,361	1
125	4	174,793	1
123	5	167,304	1
163	(Null)	187,902	1
101	(Null)	112,396	1
111	(Null)	118,205	1
111	(Null)	157,322	1
171	(Null)	178,353	1

Рисунок 5.8 – Результат виконання скрипта

5.2.2.2. 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 scholarship 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, scholarship);
            RAISE NOTICE 'Задану заяву було успішно зараховано';
        ELSE
            RAISE NOTICE 'Заданого абітурієнта вже було зараховано за іншою заявою';
        END IF;
    END IF;
```

```

        END IF;
    END;
$$ LANGUAGE plpgsql;

```

Результат:

```

CALL enroll_abit(8)
> ПОВІДОМЛЕННЯ:
                Новий абітурієнт
                ПІБ: Балицька Ірина Володимирівна
                Пошта: ol070506@gmail.com

> ПОВІДОМЛЕННЯ: SEND LETTER;
send_to=ol070506@gmail.com;
text="Шановний(на) Балицька Ірина Володимирівна\nВас зараховано на 1, Кафедра економічної кібернетики,
ФММ із вступним балом 117.660. Вам потрібно буде з'явитися в деканаті протягом тижня з дня надходження
цього листа для підписання документів.\n\nЗ повагою, Деканат";

> ПОВІДОМЛЕННЯ: Задану заяву було успішно зараховано
> OK
> Time: 0,015s

```

Рисунок 5.9 – Результат виконання скрипта

5.2.2.3. Процедури та функції декана

- Процедура, що дає можливість сформулювати 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:

```

CALL formation_of_order(2)
> ПОВІДОМЛЕННЯ: Зарахувати на Кафедра виробництва приладів наступних абітурієнтів:
Міщенко Петро Сергійович
Піддубний Мстислав
> OK
> Time: 0,002s

```

Рисунок 5.10 – Результат виконання скрипта

5.2.2.4. 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:

Message	Result 1
ІД зарахованої заяви 5 абітурієнта	13

Рисунок 5.11 – Результат виконання скрипта

5.2.3. 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:

Message	Result 1						
id	first_name	last_name	certificate_score	specialty_name	priorities	score	course
24	Міщенко	Петро	110,000	Професійна освіта (за спе	1	117,378	1
23	Микитюк	Діана	129,000	Комп'ютерна інженерія	1	130,478	1
2	Слісаренко	Богдан	162,000	Електроніка	1	132,953	1
66	Дацко	Юлія	149,000	Професійна освіта (за спе	1	139,745	1
17	Балицька	Ірина	189,000	Телекомунікації та радіотех	1	150,549	1
6	Христова	Ольга	181,000	Державна безпека	1	156,657	1
62	Чичкан	Ярослав	111,000	Державна безпека	1	166,868	1
59	Сереженко	Юрій	141,000	Телекомунікації та радіотех	1	168,587	1
36	Балінський	Ростислав	132,000	Інженерія програмного за	1	196,300	1
12	Балицька	Ірина	189,000	Науки про освіту	2	114,100	1
7	Слісаренко	Богдан	162,000	Електроніка	2	133,809	1
65	Балінський	Ростислав	132,000	Комп'ютерна інженерія	2	143,250	1
13	Дацко	Юлія	149,000	Математика	2	155,815	1

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:

id	first_name	last_name	short_name	department_name	specialty_name	certificate_score	priorities	score	course	department_priority
17	Балицька	Ірина	ФЕЛ	Кафедра мікроелектроніки	Телекомунікації та радіотехніка	189,000	1	150,549	1	1
12	Балицька	Ірина	ФЕЛ	Кафедра мікроелектроніки	Науки про освіту	189,000	2	114,100	1	2
8	Балицька	Ірина	ФІММ	Кафедра економічної кібер	Інженерія програмного забезпечення	189,000	3	117,660	1	1
78	Балицька	Ірина	ФБТ	Кафедра біотехніки та інженерії	Інженерія програмного забезпечення	189,000	3	167,841	3	1
56	Балицька	Ірина	ФПМ	Кафедра прикладної математики	Прикладна фізика та нанотехнології	189,000	4	198,118	1	2
54	Балицька	Ірина	ФЕА	Кафедра відновлюваних джерел енергії	Науки про освіту	189,000	(Null)	196,661	1	2
58	Балицька	Ірина	ФІОТ	Кафедра інформаційних систем	Електроніка	189,000	(Null)	105,434	1	2
63	Балицька	Ірина	ФПМ	Кафедра прикладної математики	Електроніка	189,000	(Null)	178,955	1	2
36	Балінський	Ростислав	ФПМ	Кафедра прикладної математики	Інженерія програмного забезпечення	132,000	1	196,300	1	1
65	Балінський	Ростислав	ІХВ	Кафедра екології та технологій	Комп'ютерна інженерія	132,000	2	143,250	1	2
55	Балінський	Ростислав	ФЛ	Кафедра української мови, Математика		132,000	3	119,361	1	1
53	Балінський	Ростислав	ФБМІ	Кафедра технологій оздоровлення	Кібербезпека	132,000	4	174,793	1	1
9	Балінський	Ростислав	ХТФ	Кафедра фізичної хімії	Комп'ютерна інженерія	132,000	5	167,304	1	2

Рисунок 5.13 – Результат виконання скрипта

5.2.4. 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:

id_subject	min
1	128,792
2	104,386
3	103,279
4	127,890
5	103,784
6	105,187
7	108,504
8	110,510

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
id_subject	max											
1	187,352											
2	195,255											
3	196,902											
4	193,662											
5	192,430											
6	188,544											
7	184,000											
8	195,779											

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
id_subject	avg											
1	162,36000000000000											
2	172,60088888888889											
3	157,32766666666667											
4	159,65783333333333											
5	167,59180000000000											
6	130,99700000000000											
7	156,91933333333333											
8	162,13742857142857											

Рисунок 5.16 – Результат виконання скрипта

- Витягнути всі паспорти

Текст скрипта:

```
SELECT *
FROM "document"
WHERE "document".short_description = 'Паспорт';
```

Результат:

id	id_abiturient	short_description	document_url
1	1	Паспорт	https://documents.com/id/
3	3	Паспорт	https://documents.com/id/
5	2	Паспорт	https://documents.com/id/
7	5	Паспорт	https://documents.com/id/
9	4	Паспорт	https://documents.com/id/
11	7	Паспорт	https://documents.com/id/
13	6	Паспорт	https://documents.com/id/
15	9	Паспорт	https://documents.com/id/
17	8	Паспорт	https://documents.com/id/
19	10	Паспорт	https://documents.com/id/
21	13	Паспорт	https://documents.com/id/
23	12	Паспорт	https://documents.com/id/
25	11	Паспорт	https://documents.com/id/

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
full_name	short_name	department_name										
Інженерно-хімічний факул	ІХВ	Кафедра екології та технол										
Приладобудівний факульт	ПБФ	Кафедра виробництва при.										
Радіотехнічний факультет	РТФ	Кафедра прикладної радіо										
Факультет біомедичної інж	ФБМІ	Кафедра технологій оздор										
Факультет біотехнології і б	ФБТ	Кафедра біотехніки та інже										
Факультет електроенергот	ФЕА	Кафедра відновлюваних дж										
Факультет електроніки	ФЕЛ	Кафедра мікроелектроніки										
Факультет інформатики та	ФІОТ	Кафедра інформаційних си										
Факультет інформатики та	ФІОТ	Кафедра інформатики та п										
Факультет лінгвістики	ФЛ	Кафедра української мови,										
Факультет менеджменту та	ФММ	Кафедра економічної кібер										
Факультет прикладної мат	ФПМ	Кафедра прикладної матем										
Факультет соціології і прав	ФСП	Кафедра психології і педаг										

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
id	certificate_score											
1	141,000											
2	173,000											
3	132,000											
4	162,000											
5	149,000											
6	191,000											
7	129,000											
8	110,000											
9	181,000											
10	200,000											
11	111,000											
12	194,000											
13	189,000											

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
course	count											
1	73											
3	5											

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
id_specialty	count											
171	12											
17	6											
101	6											
121	6											
163	6											
172	6											
251	6											
11	5											
15	5											
105	5											
111	5											
123	5											
125	5											

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:

Message	Result 1	Result 2	Result 3	Result 4	Result 5	Result 6	Result 7	Result 8	Result 9	Result 10	Result 11	Result 12
row_number	ПІБ	subject_name		result								
1	Сереженко Юрій Левович	Українська мова та літерат		193,662								
2	Сереженко Юрій Левович	Математика		178,256								
3	Сереженко Юрій Левович	Біологія		189,124								
4	Сереженко Юрій Левович	Іноземна мова		119,153								
5	Яськевич Оксана	Біологія		162,650								
6	Яськевич Оксана	Математика		183,562								
7	Яськевич Оксана	Хімія		103,279								
8	Яськевич Оксана	Історія України		181,116								
9	Балінський Ростислав Андрійович	Іноземна мова		132,725								
10	Балінський Ростислав Андрійович	Історія України		139,610								
11	Балінський Ростислав Андрійович	Фізика		103,784								
12	Слісаренко Богдан Святославович	Історія України		184,820								
13	Слісаренко Богдан Святославович	Фізика		169,449								

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:

id	concat	short_description	document_url
1	Серезенко	Паспорт	https://documents.com/id/
2	Серезенко	РНКОПП	https://documents.com/id/
3	Балінський	Паспорт	https://documents.com/id/
4	Балінський	РНКОПП	https://documents.com/id/
5	Яськевич О	Паспорт	https://documents.com/id/
6	Яськевич О	РНКОПП	https://documents.com/id/
7	Дацко Юлія	Паспорт	https://documents.com/id/
8	Дацко Юлія	РНКОПП	https://documents.com/id/
9	Слісаренко	Паспорт	https://documents.com/id/
10	Слісаренко	РНКОПП	https://documents.com/id/
11	Микитюк Д	Паспорт	https://documents.com/id/
12	Микитюк Д	РНКОПП	https://documents.com/id/
13	Роп*яник Г	Паспорт	https://documents.com/id/

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:

first_name	last_name	fathers_name	email	phone_number
Микитюк	Діана	Миколаївна	andrea183@gmail.com	+380724113914
Слісаренко	Богдан	Святославович	jchton@gmail.com	+380346227379
Дацко	Юлія	Генадійвна	echigo93@gmail.com	+380546110875
Слісаренко	Богдан	Святославович	jchton@gmail.com	+380346227379
Дацко	Юлія	Генадійвна	echigo93@gmail.com	+380546110875
Христова	Ольга	Семенівна	verstart@gmail.com	+380542831554
Слісаренко	Богдан	Святославович	jchton@gmail.com	+380346227379
Балицька	Ірина	Володимирівна	ol070506@gmail.com	+380139223551
Балінський	Ростислав	Андрійович	robgle@gmail.com	+380746425919
Слісаренко	Богдан	Святославович	jchton@gmail.com	+380346227379
Христова	Ольга	Семенівна	verstart@gmail.com	+380542831554
Балицька	Ірина	Володимирівна	ol070506@gmail.com	+380139223551
Дацко	Юлія	Генадійвна	echigo93@gmail.com	+380546110875

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:

department_name	count
Кафедра відновлюваних джерел енергії	11
Кафедра прикладної математики	9
Кафедра економічної кібернетики	7
Кафедра мікроелектроніки	7
Кафедра інформатики та програмної інженерії	7
Кафедра інформаційних систем та технологій	6
Кафедра біотехніки та інженерії	5
Кафедра української мови, літератури та культури	5
Кафедра фізичної хімії	5
Кафедра технологій оздоровлення і спорту	5
Кафедра виробництва приладів	4
Кафедра екології та технологій рослинних полімерів	4
Кафедра психології і педагогіки	3

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:

full_name	count
Факультет інформатики та обчислювальної техніки	13
Факультет електроенерготехніки та автоматики	11
Факультет прикладної математики	9
Факультет менеджменту та маркетингу	7
Факультет електроніки	7
Факультет лінгвістики	5
Хіміко-технічний факультет	5
Факультет біотехнології і біотехніки	5
Факультет біомедичної інженерії	5
Інженерно-хімічний факультет	4
Приладобудівний факультет	4
Факультет соціології і права	3

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:

subject_name	count
Фізика	9
Українська мова та література	6
Історія України	6
Математика	8
Іноземна мова	6
Географія	3
Хімія	6
Біологія	7

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:

subject_name	count
Фізика	10
Математика	9
Біологія	7
Історія України	6
Іноземна мова	6
Хімія	6
Українська мова та література	6
Географія	3

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:

concat	score
Роп'яник Ганна Арсенівна	168,891
Дацко Юлія Геннадіївна	155,815

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:

subject_name	count
Фізика	1
Математика	1

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:

id	ПІБ	certificate_s	specialty_name	priorities	score	course
36	Балінський Ростислав Андрій	132,000	Інженерія програмного забезпечення	1	196,300	1
13	Дацко Юлія Генадіївна	149,000	Математика	2	155,815	1
71	Піддубний Мстислав	200,000	Екологія	4	136,561	1
44	Яськевич Оксана	173,000	Електроніка	5	105,551	1
40	Роп'яник Ганна Арсенівна	191,000	Науки про освіту	(Null)	168,891	1
73	Міщенко Петро Сергійович	110,000	Телекомунікації та радіотехніка	(Null)	188,854	1
35	Сереженко Юрій Левович	141,000	Електроніка	(Null)	198,430	1

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:

id	specialty_name	count
171	Електроніка	2
121	Інженерія програмного за	1
101	Екологія	1
11	Науки про освіту	1
172	Телекомунікації та радіотех	1
111	Математика	1

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:

ПІБ	certificate_score	zno_res
► Сереженко Юрій Левович	141,000	170,0487500000000000
Яськевич Оксана	173,000	157,6517500000000000
Балінський Ростислав Андрійович	132,000	125,3730000000000000
Слісаренко Богдан Святославович	162,000	161,2086666666666667
Дацко Юлія Генадійвна	149,000	144,2270000000000000
Роп*яник Ганна Арсенівна	191,000	153,9375000000000000
Микитюк Діана Миколаївна	129,000	144,2690000000000000
Міщенко Петро Сергійович	110,000	187,6610000000000000
Христова Ольга Семенівна	181,000	156,1807500000000000
Піддубний Мстислав	200,000	170,3254000000000000
Чичкан Ярослав Богданович	111,000	163,6187500000000000
Косяченко Анастасія	194,000	164,6960000000000000
Балицька Ірина Володимирівна	189,000	179,2495000000000000

Рисунок 5.33 – Результат виконання скрипта

5.2.5. 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:

```

5 INSERT INTO enrolled_abiturient (id_statement, scholarship) VALUES
6 (35, FALSE),
7 (44, FALSE),
8 (36, FALSE),
9 (13, TRUE),
10 (40, TRUE),
11 (73, FALSE),
12 (71, FALSE);

```

Рисунок 5.30 – Приклад вхідних даних для випробування генератора

id	id_statement	scholarship
1	35	f
2	44	f
3	36	f
4	13	t
5	40	t
6	73	f
7	71	f

Рисунок 5.34 – Результат виконання скрипта

5.2.6. 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:

Message	Explain 1	Result 1	Result 2
<p>QUERY PLAN</p> <p>Seq Scan on statements</p> <p>Filter: (score > '150'::numeric)</p>			

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:

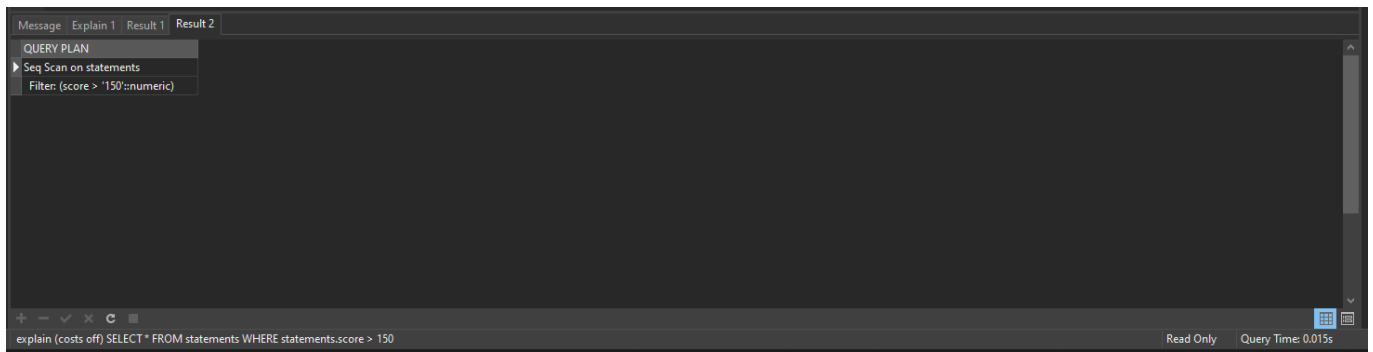


Рисунок 5.36 – Результат виконання скрипта після оптимізації

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 – PostgreSQL. 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

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

(Найменування програми (документа))

Онлайн-репозиторій

(Вид носія даних)

12 Кб, 8 аркушів

(Обсяг програми (документа), арк.,

студента групи ІІІ-13 ІІ курсу

Замкового Дмитра

Володимировича


```
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,
'robgle@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, 'РНКOPP', 'https://documents.com/id/2'),
(3, 3, 'Паспорт', 'https://documents.com/id/3'),
(4, 3, 'РНКOPP', 'https://documents.com/id/4'),
(5, 2, 'Паспорт', 'https://documents.com/id/5'),
(6, 2, 'РНКOPP', 'https://documents.com/id/6'),
(7, 5, 'Паспорт', 'https://documents.com/id/7'),
(8, 5, 'РНКOPP', 'https://documents.com/id/8'),
(9, 4, 'Паспорт', 'https://documents.com/id/9'),
(10, 4, 'РНКOPP', 'https://documents.com/id/10'),
(11, 7, 'Паспорт', 'https://documents.com/id/11'),
(12, 7, 'РНКOPP', 'https://documents.com/id/12'),
(13, 6, 'Паспорт', 'https://documents.com/id/13'),
(14, 6, 'РНКOPP', 'https://documents.com/id/14'),
(15, 9, 'Паспорт', 'https://documents.com/id/15'),
(16, 9, 'РНКOPP', 'https://documents.com/id/16'),
(17, 8, 'Паспорт', 'https://documents.com/id/17'),
(18, 8, 'РНКOPP', 'https://documents.com/id/18'),
(19, 10, 'Паспорт', 'https://documents.com/id/19'),
(20, 10, 'РНКOPP', 'https://documents.com/id/20'),
(21, 13, 'Паспорт', 'https://documents.com/id/21'),
(22, 13, 'РНКOPP', 'https://documents.com/id/22'),
(23, 12, 'Паспорт', 'https://documents.com/id/23'),
(24, 12, 'РНКOPP', 'https://documents.com/id/24'),
(25, 11, 'Паспорт', 'https://documents.com/id/25'),
(26, 11, 'РНКOPP', '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);
```