

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 для підтримки екзаменаційної та залікової сесії, поточного та календарного контролів

Студента(ки) 2 курсу ІІ-22 групи спеціальності  
121 “Інженерія програмного забезпечення”

Монтицького Назарія Олеговича

(прізвище та ініціали)

Керівник ст. вик. Марченко Олена Іванівна

(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)

**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-22 Семестр 3

**З А В Д А Н Н Я**  
**НА КУРСОВУ РОБОТУ СТУДЕНТУ**

Монтицькому Назарію Олеговичу  
(прізвище, ім'я, по-батькові)

Тема роботи База даних для підтримки екзаменаційної та залікової сесії, поточного та календарного контролів

Керівник роботи Марченко Олена Іванівна  
(прізвище, ім'я, по-батькові)

1. Строк подання студентом роботи 29.12.2023

Вихідні дані до роботи Необхідність розробити якісну концептуальну модель та надійну фізичну інфраструктуру для проведення екзаменаційної та залікової сесії, поточного та календарного контролів

2. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити) \_\_\_\_\_

1) Аналіз предметного середовища

2) Побудова ER-моделі

3) Побудова реляційної схеми з ER-моделі

4) Створення бази даних, у формі 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

3. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

4. Дата видачі завдання 08.11.2023

## КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання курсового проєкту	Строк виконання етапів проєкту	Примітка
1	Аналіз предметного середовища	25.11.2023	
2	Побудова ER-моделі	30.11.2023	
3	Побудова реляційної схеми з ER-моделі	02.12.2023	
4	Створення бази даних, у форматі обраної системи управління базою даних	05.12.2023	
5	Створення користувачів бази даних	07.12.2023	
6	Імпорт даних з використанням засобів СУБД в створену базу даних	08.12.2023	
7	Створення мовою SQL запитів	10.12.2023	
8	Оптимізація роботи запитів	12.12.2023	
9	Оформлення пояснювальної записки	15.12.2023	
10	Захист курсової роботи	30.12.2023	

Студент

\_\_\_\_\_  
(підпис)

Монтицький Н. О.

\_\_\_\_\_  
(прізвище та ініціали)

Керівник роботи

\_\_\_\_\_  
(підпис)

ст. вик. Марченко О. І.

\_\_\_\_\_  
(прізвище та ініціали)

## ЗМІСТ

ЗМІСТ .....	3
1 ВСТУП.....	5
2 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА .....	6
3 АНАЛІЗ ПРОГРАМНИХ ПРОДУКТІВ .....	14
4 ПОСТАНОВКА ЗАВДАННЯ .....	16
5 ПОБУДОВА ER-МОДЕЛІ .....	18
6 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ .....	28
1. Створення таблиць засобами SQL .....	29
2. Створення обмежень бази даних засобами SQL .....	38
3. Створення користувачів бази даних .....	41
4. ER-діаграма створена засобами СУБД .....	42
7 РОБОТА З БАЗОЮ ДАНИХ .....	44
1. Генератори .....	44
2. Збережені процедури та функції .....	44
3. Тригери .....	57
4. Представлення .....	65
5. SQL-запити .....	68
6. Оптимізація запитів .....	88
8 ВИСНОВОК .....	92
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ .....	93
ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ТАБЛИЦЬ ....	94
ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ОБМЕЖЕНЬ	97
ДОДАТОК В ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ТРИГЕРІВ ....	98
ДОДАТОК Д ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ФУНКЦІЙ ТА	
ЗБЕРЕЖЕНИХ ПРОЦЕДУР .....	103
ДОДАТОК Е ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ	
ПРЕДСТАВЛЕНЬ .....	111

ДОДАТОК Ж	ТЕКСТИ ПРОГРАМНОГО КОДУ ЗАПИТІВ .....	113
ДОДАТОК З	ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ІНДЕКСІВ ..	121
ДОДАТОК И	ТЕКСТИ ПРОГРАМНОГО КОДУ ІМПОРТУ ДАНИХ .....	122
ДОДАТОК К	СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ .....	125

## 1 ВСТУП

Курсова робота з дисципліни “Бази даних” — це освітній компонент, що перевіряє фахові компетентності та навички здобувачів освіти у предметній області ефективного оперування даними та вмілого менеджменту інформації.

Темою цієї курсової роботи є “База даних для підтримки екзаменаційної та залікової сесії, календарного та семестрового контролів”. У ході виконання проєкту будуть розкриті основні аспекти регулювання навчального процесу у закладі вищої освіти на площині оперування даними за допомогою надбань науково-технічного прогресу.

Бази даних є досить ефективним засобом оперування великих об’ємів інформації. Курсовий проєкт надасть можливість вдало продемонструвати набуті навички протягом семестру, виконуючи імплементацію теми роботи у фізичну базу даних засобами обраної системи керування базою даних.

Тема курсової роботи є досить релевантною, оскільки вища освіта здобувається багатьма людьми по всьому світу. Відомості про успішність студентів, викладачів та перебіг навчального процесу мають досить великий обсяг, тому їх використання на папері є марнотратним та неефективним у використанні.

У ході виконання курсової роботи будуть розкриті основні аспекти аналізу предметної області, проєктування концептуальної моделі майбутньої бази даних та перехід до розробки інфраструктури фізичної БД.

## 2 АНАЛІЗ ПРЕДМЕТНОГО СЕРЕДОВИЩА

Предметна область семестрового та календарного контролів, екзаменаційної та залікової сесії охоплює досить широкий спектр понять та тверджень. Необхідно дослідити та розкрити всі потрібні постановки задачі, щоб чітко та точно оцінити середовище розробки бази даних для цієї курсової роботи. У ході дослідження галузі проведення контролю здобувачів освіти протягом навчального семестру виконаємо декомпозицію задачі для виявлення атомарних елементів предметної області, що стануть основою розробки сутностей бази даних і утворення зв'язків між ними [9].

Вищий навчальний заклад, у якому проводяться відповідні типи контролів, що зазначені вище, поділяється на його структурні підрозділи. Такими структурними підрозділами можуть бути факультети та науково-навчальні інститути. Наприклад, у Київському політехнічному інституті функціонують факультети (наприклад, Факультет інформатики та обчислювальної техніки) та інститути (наприклад, Інститут прикладного системного аналізу). З цього слідує, що кожний структурний підрозділ має свій тип, яким характеризується його вектор роботи. Також загальновідомим фактом є те, що структурні одиниці університетів мають власний веб-ресурс, мобільний номер телефону (наприклад, гаряча лінія або служба підтримки), назву, електронну пошту, тощо [9]. На основі цих виявлених фактів, ми можемо скласти орієнтовний план, який стане фундаментом розробки таблиці для сутності “Структурний підрозділ” у базі даних:

1. ID (унікальний ідентифікатор);
2. Назва;
3. Тип структурного підрозділу (зовнішній ключ);
4. Номер мобільного телефону;
5. Електронна пошта;
6. Адреса веб-ресурсу;

Виникає завдання визначення типу структурного підрозділу. Доречним є створення додаткової сутності “Тип структурного підрозділу”, у таблицю якого внесемо відомості про структурну одиницю для її подальшої характеристики. Атрибути такої таблиці будуть наступні елементи:

1. ID (унікальний ідентифікатор);
2. Значення типу структурної одиниці (наприклад, “Факультет”);

Кожний структурний підрозділ має власні відносно автономні структурні одиниці, що функціонують у його складі та виконують дослідження у їх предметних областях. Мова йде про кафедри, які є частинами факультетів та інститутів. Факультет може мати декілька кафедр, отже кафедра має обов’язково зберігати посилання на структурний підрозділ, у складі якого вона існує. Кожна кафедра має свою назву відповідно до предметної області, у якій вона проводить основну частину своїх досліджень. Можемо визначити аналогію між кафедрою та факультетом (чи інститутом) про наявність власної електронної пошти, веб-ресурсу, номеру мобільного телефону, тощо. Кафедра є досить важливим елементом цієї предметної області, оскільки студенти, їх академічні групи, викладачі, всі види контролів закріплюються за ними [9]. Тому розробка цієї сутності є досить важливою. Визначимо основні атрибути таблиці сутності “Кафедра”, що увійдуть в нашу базу даних:

1. ID (унікальний ідентифікатор);
2. Назва;
3. Структурний підрозділ (зовнішній ключ);
4. Електронна пошта;
5. Адреса веб-ресурсу;
6. Номер мобільного телефону;

Кафедра має своїх штатних працівників, які виконують всю потрібну роботу для функціонування цієї ж кафедри. Зокрема, мова йде про викладачів, які є науково-педагогічними працівниками, що здійснюють наукові дослідження в межах своєї структурної одиниці та викладають різні дисципліни для здобувачів освіти, тобто здійснюють педагогічну діяльність. Викладач у



межах кафедри має власне наукове звання, що змінюється пропорційно його науковим внескам. Науково-педагогічний працівник також характеризується своїм стажем роботи, що визначається датою початку його працевлаштування. Підсумовуючи факти, ми можемо чітко зазначити, що майбутня сутність викладача матиме атрибути, що характеризують його особу, структурну одиницю, в якій він працює, дату початку роботи та наукове звання. Виникає запитання чому саме дата початку роботи? Оскільки дані про стаж роботи потрібно оновлювати кожного року (якщо мова йде про кількість років), то це є досить не зручним [9]. Набагато зручнішим буде збереження дати початку роботи викладача і підрахунок стажу його роботи за допомогою відповідних запитів, що підтримуються системою управління базою даних (далі — СУБД).

Виділимо основні атрибути таблиці сутності “Викладач”:

1. ID (унікальний ідентифікатор);
2. Прізвище;
3. Ім’я;
4. По-батькові;
5. Кафедра (зовнішній ключ);
6. Науковий ступінь (зовнішній ключ);
7. Початок роботи (дата);

Науковий ступінь доречно винести у окрему таблицю. Це є доречним оскільки науковий ступінь може бути однаковим у багатьох викладачів, тому відповідні назви ступенів можемо зберегти у іншій таблиці, на рядок якої можемо посилатися. Приблизні атрибути таблиці “Науковий ступінь” виглядатиме ось так:

1. ID (унікальний ідентифікатор);
2. Науковий ступінь (символьний тип);

Перед тим як почати опис студента та плану його реалізації у базі даних, потрібно виокремити одне особливе твердження, яке слід розглянути, в ході аналізу цієї предметної області. Зокрема, мова йде про академічну групу, до якої належать студенти. Академічна група — це група здобувачів освіти, що

об'єднана в межах однієї освітньої програми за певною ознакою. Студентів набагато практичніше об'єднати у певні групи, щоб спростити пошук та доступ до них. Слід зазначити, що студент закріплюється за певною кафедрою саме через групу, до якої він належить. Тобто, якщо умовна група “ІІ-22” функціонує в межах Кафедри інформатики та програмної інженерії, то і студенти, які вчаться у цій групі належать до тієї ж кафедри. Для кожної академічної групи закріплюється її академічний куратор, який за сумісництвом є викладачем і не обов'язково працює на тій же кафедрі. З цього опису академічної групи можемо скласти її відносну структуру, що ляже в основі атрибутів таблиці її сутності:

1. ID (унікальний ідентифікатор);
2. Назва;
3. Кафедра (зовнішній ключ);
4. Академічний куратор (зовнішній ключ);

Перейдемо до дослідження такого елементу цієї предметної області як здобувач освіти або студент. Здобувач освіти виконує свої обов'язки, перебуваючи у межах певної академічної групи. Це твердження утворює логічний зв'язок “Студент - Група - Кафедра”. При розробці сутності студента слід вказати у його атрибутах відомості, що характеризують його як людину. Конкретно здобувача освіти характеризує дата вступу до університету, що дозволить визначити курс на якому студент навчається, використовуючи засоби СУБД [9]. З вищенаведених фактів маємо можливість скласти орієнтовну структуру таблиці, до якої увійдуть персональні дані студента, його група та дата вступу. Отже, сутність “Студент” матиме наступні атрибути:

1. ID (унікальний ідентифікатор);
2. Прізвище;
3. Ім'я;
4. По-батькові;
5. Група (зовнішній ключ);
6. Дата вступу;

Характеристика проведення певного типу контролю потребує додаткової декомпозиції задачі для виявлення додаткових тверджень, що ляжуть в основі розробки бази даних.

Контроль здобувачів освіти проводиться по певному предмету. Студенти протягом семестру здобувають загальні та фахові компетентності та підтверджують їх наявність протягом контрольних заходів. Предмет, по якому студент складає контроль, має свою певну назву, що характеризує предметну область, що викладається студентам. Освітньою програмою передбачено кількість годин Європейської кредитної трансферно-накопичувальної системи (далі — ЄКТС), що визначають кількість занять з цього предмету протягом семестру. Важливим аспектом також є складання семестрового контролю з даного предмету, тобто чи є цей предмет заліком, чи є екзаменом. Це визначає спосіб складання предмету [9]. Тип предмету доречно винести в окрему сутність, де сутність предмету посилатиметься на неї. З цього опису можемо скласти орієнтовну структуру атрибутів потенційної сутності “Предмет”:

1. ID (унікальний ідентифікатор);
2. Назва;
3. Тип предмету (зовнішній ключ);
4. Кількість годин;

Таблиця “Тип предмету” буде представлена у базі даних з метою характеристики навчальної дисципліни, що викладається. У цій таблиці будуть зберігатися дані з множини {“Екзамен”, “Залік”, “Курсова робота”}, можливо інші типи, якщо вони з’являться. Сутність “Тип предмету” матиме орієнтовно наступну структуру:

1. ID (унікальний ідентифікатор);
2. Назва типу предмету;

За кожний вид контролю, який складає студент з певної дисципліни, виставляється оцінка. Цей елемент предметної області є не таким простим, як здається на перший погляд. Студент може отримати певну кількість балів за окремий контрольний захід. Ця кількість балів має задовільняти стобальну

систему, функціонування якої ухвалено Болонською системою освіти. Бали можуть бути “конвертована” у оцінку за національною шкалою (наприклад, “Відмінно”). Такий принцип розповсюджується на систему оцінювання ЄКТС, яка виражається у літерах (наприклад, “А”). Тобто, сутність “Оцінка” матиме наступну структуру:

1. ID (унікальний ідентифікатор);
2. Кількість балів;
3. Національна оцінка;
4. Оцінка за ЄКТС;

Кожний контроль має тип за яким визначається спосіб його проведення. Тобто, це може бути календарний або семестровий контроль, залікова або екзаменаційна сесія. Доречним буде характеризувати контроль за допомогою окремої сутності [9]. На кожний вид контролю відводиться певна кількість годин. Розробимо новий об’єкт “Тип контролю”, яка матиме наступну структуру:

1. ID (тип контролю);
2. Назва типу контролю;
3. Кількість годин.

Тепер слід описати ключову сутність розробки цієї бази даних, а саме контроль. Контроль буде визначатися своїм типом, за яким він проводиться. У нашій базі даних ця сутність буде визначати весь процес оцінювання конкретного студента за конкретним контрольним заходом з певного предмету. З цих фактів слідує, що одним з атрибутів цього об’єкту буде зовнішній ключ на конкретного здобувача освіти, над знаннями якого проводиться контроль. Оскільки знання оцінюються з певної навчальної дисципліни, то сутність посилатиметься також на конкретний предмет. Таблиця “Контроль” матиме зовнішній ключ на викладача, який оцінюватиме студента. Оскільки є різні види контролів, то можна і зв’язати різних викладачів на один і той самий предмет. Знання студента оцінюються певною мірою, а саме оцінкою, на яку контроль теж має посилання. Щоб закріпити за студентом статус його атестації

додамо булевий атрибут, що вказує на позитивний результат (наприклад, більше 60% від обсягу виконаних вимог). Одним з визначальних атрибутів сутності контролю буде дата його проведення. Цей атрибут є більш корисним ніж здається на перший погляд. Наприклад, нам потрібно дізнатися результат атестації студента під час першого календарного контролю. Ми знаємо, що календарний контроль проводиться, починаючи з дати “А”, закінчуючи датою “В”. За допомогою засобів СУБД, можемо створити запит про вибірку даних по контролю конкретного студента у період (А; В) [9]. Отже, на основі наявних фактів, які ми провели у ході аналізу предметного середовища, можемо скласти структуру сутності контролю, яка буде спроектована у базі даних. Об’єкт матиме наступні атрибути:

1. ID (унікальний ідентифікатор);
2. Тип контролю (зовнішній ключ);
3. Оцінка (зовнішній ключ);
4. Студент (зовнішній ключ);
5. Предмет (зовнішній ключ);
6. Викладач (зовнішній ключ);
7. Атестація (булеве значення);
8. Дата проведення контролю;

База даних не обмежується лише наявністю таблиць та зв’язків між ними. Нам слід визначити та описати права, ролі та користувачів, які функціонуватимуть у базі даних.

Студент, як учасник освітнього процесу, має знати результати оцінювання його знань. Тому він може мати безпосередній доступ до перегляду даних, про його успішність. Це стосується таблиці контролю та можливості вибірки певних даних з неї. Зокрема це стосується даних про предмет, викладача, оцінку, тощо [9].

Викладач теж є учасником процесу оцінки знань студента. Він має право на внесення даних про певного студента. Тобто додавання, видалення та вибірку даних із таблиці контролю [9].

Щодо інших таблиць, то доступ до них визначається іншими ролями. Слід обговорити таку роль як працівник кафедри. Він може визначати предмети, що викладаються на цій кафедрі, додавання і видалення студентів з певних груп, додавання і видалення викладачів з бази даних. Тобто на працівника кафедри мають лягати безпосередньо ті обов'язки, що впливають на роботу цієї кафедри [9].

Доцільним є створення цілі “Працівник університету”, який вже зможе визначати типи контролів, типи оцінок, наукові звання, типи підрозділів, тощо. Тобто ця роль відповідатиме за питання, що впливають на функціонування всіх структурних одиниць та кафедр у вищій навчальній установі.

### 3 АНАЛІЗ ПРОГРАМНИХ ПРОДУКТІВ

Тема розробки бази даних цієї курсової роботи передусім має бути релевантною та практичною у застосуванні. База даних для проведення екзаменаційної та залікової сесії, семестрового та календарного контролю корисна зокрема для закладів вищої та середньо-технічної освіти, яким було би зручно проводити всі заходи з використанням технологічних засобів замість паперових журналів та залікових книжок.

Якщо створити якісне програмне забезпечення, що супроводжується грамотно створеною базою даних, що підтримує цілісність та логічність даних, то освітній процес можна зробити ефективнішим по часу та витратам. Слід визначити, які вже наявні програмні продукти існують, що виконують подібну роль.

Передусім слід виокремити “Електронний кампус” (далі — Кампус), що використовується в КПІ ім. Ігоря Сікорського (далі — Університет), для ведення всіх типів контрольних заходів. Кожний студент будь-якого підрозділу має власний обліковий запис у цій системі та має можливість переглядати результати навчання по предметам, які він вивчає на даний момент. Це можуть бути як результати оцінювання лабораторних робіт, так і будь-які модульні контрольні роботи або навіть заліки та екзамени [2].

Доступ до Кампусу також мають викладачі, що читають дисципліни та проводять окремі контрольні заходи. Науково-педагогічний працівник може знайти студента у системі за допомогою його групи та імені. Звісно, що викладач має право виставити студенту оцінку за контрольний захід, а також в має можливість визначити який саме вид контролю оцінюється.

Гарним прикладом подібної системи оцінювання успішності здобувачів освіти є Moodle. Система організована для зручності як і викладачів, так і студентів. Цим програмним продуктом користуються багато навчальних закладів і Університет не є виключенням. Зокрема, як приклад можна привести

Національний університет водного господарства і природокористування (м. Рівне), Національний університет “Острозька академія” (м. Острог), тощо [3].

Студент має зручний інтерфейс, де він може перейти до сторінки будь-якої дисципліни і переглянути всі доступні документи, результати власного оцінювання у розділі “Журнал оцінок”, тощо. Викладачі теж мають об’ємний діапазон функціоналу, що дозволяє їм досить вільно регулювати освітній процес. Звісно, що все програмне забезпечення підтримується досить потужною базою даних, яка дозволяє вміщувати в собі великі об’єми інформації.

У закладах середньої освіти(школи, гімназії, ліцеї, тощо) розпочали перехід від паперових відомостей(класні журнали, таблиці, тощо) до електронних застосунків, що дозволяють учасникам освітнього процесу: вчителі, учні та батьки — регулювати та відстежувати хід навчання. Прикладом такого програмного забезпечення є додаток “Human”. Програмне забезпечення дозволяє учням завантажувати домашнє завдання, переглядати навчальні матеріали, в свою чергу батьки мають доступ до перегляду успішності своїх учнів, замість того, щоб окремо звертатися до вчителя.

Викладачі за допомогою “Human” можуть виставляти оцінки, завантажувати навчальні матеріали, регулювати освітній процес іншими засобами, які підтримує програмне забезпечення. Використання такого програмного забезпечення підтримується базою даних, яка містить в собі всю необхідну інформацію, що використовується в ході освітнього процесу [4].

З вищенаведеного можемо констатувати, що будь-яке програмне забезпечення, що використовується для оцінки компетентностей здобувачів освіти, має зберігати всю інформацію про хід цього процесу. Функціонал таких додатків має охоплювати можливості збереження різноманітних файлів та документації, що може використовуватися під час навчання.



#### 4 ПОСТАНОВКА ЗАВДАННЯ

У розділах “Опис предметного середовища” та “Аналіз програмних продуктів” ми провели концептуальну оцінку майбутньої бази даних, що буде створена у ході проєктування логічної моделі та створення фізичної моделі.

База даних теми цієї курсової роботи охоплюватиме створення дванадцяти таблиць, що міститимуть всю релевантну інформацію про освітній процес університету в якому проводяться контрольні заходи. Імплементація сутностей у таблиці бази супроводжуються формуванням інфраструктури, що міститиме дані про структурні підрозділи університету, кафедри, студентів, академічні групи, викладачів, предмети та контрольні заходи.

Інфраструктура бази даних формується на основі ER-моделі, яку слід розробити згідно з аналізом предметного середовища. Діаграма зв'язків між сутностями забезпечить подальше розуміння розробки фізичної бази даних та імпортування інформації у неї. Діаграма забезпечена логічними зв'язками між сутностями, що послугує для формування правил у фізичній базі із використанням засобів системи управління базою даних (далі — СУБД).

Необхідно сформулювати достатньо логічні запити, які відповідають завданні і меті бази даних. Запити на отримання інформації мають нести чітку суть і завдання з якою вони формуються. Вибірка даних має відбуватися відповідно до бізнес-правил сформованих у розділі “Опис предметного середовища”.

Створення бази даних вимагає розробки відповідного функціоналу на основі збережених процедур і функцій. Процедурне забезпечення бази дозволить ефективно вилучати додаткову інформацію без створення та написання нових запитів, оскільки відповідні вибірки даних відбуватимуться всередині функцій.

Оскільки дані в базі мають бути логічними, то інфраструктуру слід забезпечити відповідними тригерами, що реагуватимуть на вставку даних і перевірятимуть валідність вхідної інформації.

Деякі запити можуть працювати довше за очікуваний час роботи. Тому слід забезпечити базу даних оптимізацією запитів через доступні засоби СУБД, аби зменшити теоретичний та практичний час виконання вибірки інформації.

Робота з базою даних включатиме в себе розробку певних представлень, які будуть зберігати інформацію про певний кластер даних, що може використовуватися в ході роботи. Представлення можуть бути корисними для різних видів користувачів бази даних.

Згідно з описом предметного середовища необхідно створити користувачів бази даних. Передусім потрібно створити ролі, яким будуть надані права на певні DML-запити. Після цього відбудеться створення користувачів, яким будуть надані ролі.

## 5 ПОБУДОВА ER-МОДЕЛІ

У розділі “Опис предметного середовища” було наведено основні відомості про сутності, які будуть реалізовані у ER-моделі. Ця модель представляється відповідною діаграмою зв’язків між сутностями, яка слугує основою побудови фізичної бази даних.

Для теми цієї курсової роботи однією з характерних сутностей є “Структурний підрозділ”, у якому проводяться контрольні заходи. Зокрема ця сутність має атрибути унікального ідентифікатора, назви, контактних даних та зовнішнього ключа, що посиляється на ідентифікатор. Ми можемо описати цю сутність у наступному представленні, що можна побачити у таблиці 1.

Таблиця 1 — Сутність структурного підрозділу університету

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Назва	Строковий параметр	
Мобільний телефон	Строковий параметр	
Електронна пошта	Строковий параметр	
Веб-ресурс	Строковий параметр	
Тип підрозділу	Цілочисельний параметр	Зовнішній ключ

У абзаці вище ми згадали про тип структурного підрозділу. Як описано у аналізі предметного середовища, таку структуру було винесено в окрему сутність, на яку посилатиметься сутність “Структурний підрозділ”. Загалом ця сутність буде простою за своєю будовою, проте важливою для логічної та концептуальної моделі бази даних.

Відображення будови сутності “Тип структурного підрозділу” сформуємо у таблиці 2.

Таблиця 2 — Сутність типу структурного підрозділу

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Тип підрозділу	Строковий параметр	

Кожний структурний підрозділ університету або іншого навчального закладу в своєму складі має атомарні підрозділи, що функціонують в його складі. У реалізації цієї бази даних це будуть кафедри, що в сукупності складають структурний підрозділ. У аналізі предметного середовища було наведено, що кафедри також мають свою назву, контактні дані, а також зовнішній ключ, де вони посилаються на структурний підрозділ, якому вони належать.

Оціночну структуру сутності “Кафедра” відобразимо у таблиці 3.

Таблиця 3 — Сутність “Кафедра” у табличному представлені

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Назва	Строковий параметр	
Мобільний телефон	Строковий параметр	
Електронна пошта	Строковий параметр	
Веб-сайт	Строковий параметр	
Структурний підрозділ	Цілочисельний параметр	Зовнішній ключ

Кафедри у складі структурного підрозділу функціонують за рахунок їх працівників, які проводять наукові дослідження, провадять наукову та освітню діяльність. Такими працівниками є викладачі, які характеризуються стажем своєї роботи та вченим званням, якого вони досягли.

У таблиці 4 сформуємо орієнтовну структуру сутності викладача кафедри, де наведемо головні атрибути.

Таблиця 4 — Сутність викладача кафедри

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Прізвище	Строковий параметр	
Ім'я	Строковий параметр	
По-батькові	Строковий параметр	
Дата влаштування	Дата	
Кафедра	Цілочисельний параметр	Зовнішній ключ
Вчене звання	Цілочисельний параметр	Зовнішній ключ

Характерним для викладача атрибутом є вчене звання, яке винесено в окрему сутність. За будовою сутність є досить проста, але для дотримання форм нормалізації баз даних, було ухвалено рішення винести вчене звання в окрему сутність.

Сформуємо орієнтовну структуру сутності “Вчене звання” у таблиці 5.

Таблиця 5 — Структура сутності “Вчене звання”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Вчене звання	Строковий параметр	

На кафедрі навчаються студенти, де вони здобувають освіту за обраною освітньою програмою. Проте слід зазначити, що студенти об'єднані за певною ознакою, а саме в академічних групах. Ці групи характеризуються назвою,

кафедрою в складі, якої вони функціонують, академічним куратором (викладач, який регулює функціонування групи студентів).

У таблиці 6 наведено приблизну структуру сутності академічної групи.

Таблиця 6 — Будова сутності академічної групи

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Назва	Строковий параметр	
Кафедра	Цілочисельний параметр	Зовнішній ключ
Академічний куратор	Цілочисельний параметр	Зовнішній ключ

Учасниками освітнього процесу, в ході якого ці учасники здобувають оцінки їх знань, є студенти, що характеризуються академічною групою в складі якої вони навчаються. Як будь-яка особа здобувач освіти має персональні дані: прізвище, ім'я і по-батькові — , а також дату коли вони вступили до університету.

У таблиці 7 наведено будову сутності “Студент”, яка буде імплементована у базу даних засобами СУБД.

Таблиця 7 — Структура сутності “Студент”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Прізвище	Строковий параметр	
Ім'я	Строковий параметр	
По-батькові	Строковий параметр	
Дата вступу	Дата	
Академічна група	Цілочисельний параметр	Первинний ключ

У межах освітніх програм здобувачі освіти формують та розвивають свої компетентності, навички та знання у певних предметних областях відповідно до предметів, які викладаються на кафедрі. Оскільки існують різні предмети, що мають різні характеристики, то виникає необхідність побудови нової сутності. Дисципліна характеризується кількістю годин, чи є цей предмет заліком чи ні та кількістю академічних годин (або кредитів ЄКТС), що на цей предмет відводиться.

З вищенаведених даних можемо побудувати табличну структуру сутності “Предмет” у таблиці 8.

Таблиця 8 — Будова сутності “Предмет”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Кількість годин	Чисельний параметр	
Тип предмету	Цілочисельний параметр	Зовнішній ключ

У таблиці 8 можемо побачити цікаву характеристику, а саме “Тип предмету”. Всі предмети поділяються на екзаменаційні та залікові. Курсова робота є окремим видом кредитних модулів, проте це також є залікова освітня компонента. Тому тип предмету можемо відобразити у табличній структурі, що наведена у таблиці 9.

Таблиця 9 — Структура сутності “Тип предмету”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Тип предмету	Строковий параметр	

По кожному предмету проводяться контрольні заходи, за які виставляються оцінки, що характеризують компетентності студента, час

виконання його роботи та відображають рівень його знань. Оцінка може виставлятися з предмету, тоді вона буде результируючою. Такі оцінки мають різні види в залежності від кількості балів у шкалі “0-100”. Наприклад, національна оцінка, яка приймає значення з множини {“Відмінно”, “Дуже добре”, “Добре”, “Задовільно”, “Достатньо”, “Не задовільно”}, або оцінки в шкалі ЄКТС, які виставляються в діапазоні “A-F”. Якщо оцінка виставляється з лабораторної роботи або іншого не підсумкового контрольного заходу, то вона не є результируючою і тому можемо її вважати такою, що не впливає на результат всього предмету.

Створимо структуру сутності “Оцінка” та відобразимо її атрибути у таблиці 10.

Таблиця 10 — Структура сутності “Оцінка”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Бали	Чисельний параметр	
Національна шкала	Строковий параметр	
Шкала ЄКТС	Символьний параметр	
Результируюча оцінка	Булеве значення	

Кожний контроль має свій тип, відповідно до якого він проводиться. Це може бути як модульна контрольна робота, так і екзамен. Можливо це також календарний або семестровий контроль, за які оцінки не виставляються, а у відомості ставляться дані про атестацію.

Тип контролю можемо охарактеризувати таблицею 11.

Таблиця 11 — Будова сутності “Тип контролю”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ



Продовження таблиці 11

Назва атрибуту	Тип атрибуту	Тип ключа
Тип контролю	Строковий параметр	
Кількість годин	Чисельний параметр	

Головною таблицею цієї бази даних є сутність, яка визначає проведення будь-якого типу контролю. Ця таблиця зберігатиме всі дані про будь-які контролі, які проводилися. Зокрема це інформація про предмет, по якому проводився контроль, тип контролю, студента, який складав контрольний захід, викладача, який проводив оцінювання, якщо це семестровий або календарний контроль, то відомості про атестацію, а також дату проведення контролю.

Відобразимо словесну структуру сутності “Контроль” у таблиці 12.

Таблиця 12 — Структура сутності “Контроль”

Назва атрибуту	Тип атрибуту	Тип ключа
Унікальний ідентифікатор	Цілочисельний параметр	Первинний ключ
Дата контролю	Дата	
Атестація	Булеве значення	
Тип контролю	Цілочисельний параметр	Зовнішній ключ
Оцінка	Цілочисельний параметр	Зовнішній ключ
Предмет	Цілочисельний параметр	Зовнішній ключ
Викладач	Цілочисельний параметр	Зовнішній ключ
Студент	Цілочисельний параметр	Зовнішній ключ

Слід визначити типи зв'язків між сутностями, які описані вище. Зв'язки є обов'язковим елементом, оскільки визначають розподілення атрибутів у зовнішніх ключах.

Між таблицями “Структурний підрозділ” та “Тип підрозділу” буде встановлений зв’язок “від одного до багатьох”, оскільки структурний підрозділ може мати лише один тип, проте тип підрозділу може бути присвоєний різним структурним підрозділам університету.

Між сутностями “Структурний підрозділ” та “Кафедра” встановимо зв’язок “Від одного до багатьох”, оскільки структурний підрозділ у своєму складі може мати кількість кафедр більшу за одиницю, проте кафедра входить до складу лише одного структурного підрозділу.

Сутності “Група” та “Кафедра” поєднаємо зв’язком “від одного до багатьох”, оскільки група може входити лише до однієї кафедри, а кафедра може мати велику кількість академічних груп у своєму складі.

Сутності “Кафедра” і “Викладач” поєднуються зв’язком “від одного до багатьох”, оскільки викладач працює в складі лише однієї кафедри, проте кафедра в своєму складі має велику кількість викладачів, що працюють на ній.

Такі сутності як “Група” та “Викладач” (де викладач функціонує в якості академічного куратора) поєднуємо зв’язком “один до одного”, оскільки у групи може бути лише один академічний куратор, а викладач може бути куратором лише однієї групи.

“Викладач” з “Вченим званням” поєднуються зв’язком “один до багатьох”, оскільки викладач може володіти лише одним вченим званням, проте вчене звання може бути присвоєне для багатьох викладачів.

Зв’язок “один до багатьох” буде поєднувати сутності “Студент” та “Академічна група”, оскільки студент може входити лише до складу однієї групи, проте академічна група в своєму складі містить кількість студентів, що більша за одиницю.

Зв'язком “один до багатьох” поєднаємо сутність “Контроль” та сутності “Студент”, “Тип контролю”, “Предмет”, “Викладач”, “Оцінка”. Чому саме так? Конкретний контроль може проводитися лише для одного студента, проте студент може проходити велику кількість контролів. Контроль може мати певний конкретний тип, проте тип контролю може бути присвоєний для багатьох контролів. Контроль проводиться по конкретному предмету, проте предмет включає в себе різні контролі (лабораторні роботи, комп'ютерні практикуми, контрольні роботи). За контроль може бути виставлена лише певна конкретна оцінка, проте одна й та сама оцінка може бути виставлена за різні контролі.

Між сутностями “Предмет” та “Тип предмету” встановлюємо зв'язок “один до багатьох”, оскільки предмет може мати лише один тип, проте один і той самий тип предмету може бути присвоєний для багатьох предметів.

Формування словесного опису ER-моделі дозволяє нам побудувати графічну діаграму, яка є проекцією фізичної моделі бази даних. За допомогою ER-діаграми відбудеться написання програмного коду створення бази даних.

Діаграму відношень між сутностями відображено на рисунку 1.

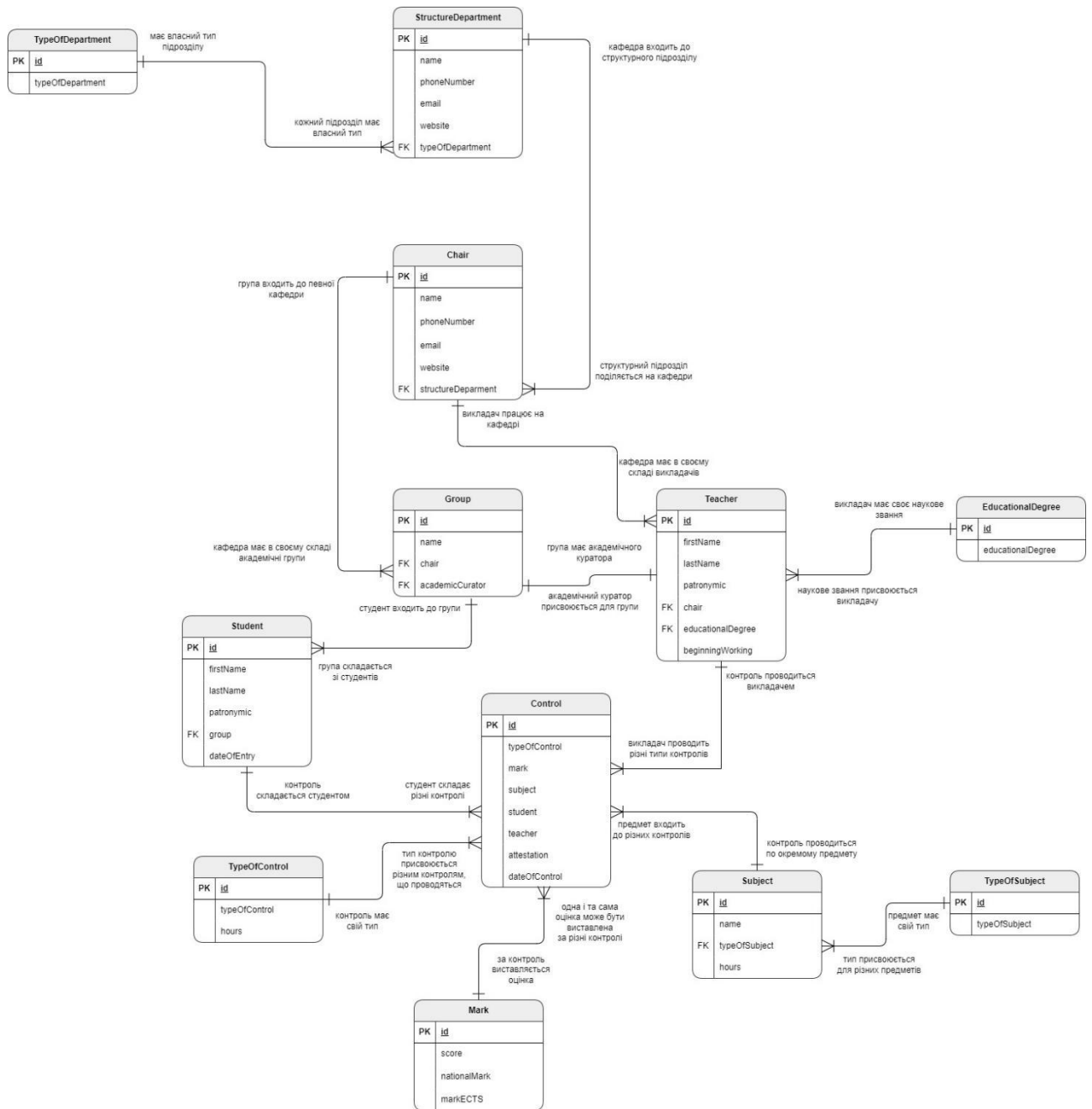


Рисунок 1 — ER-діаграма бази даних

## 6 РЕАЛІЗАЦІЯ БАЗИ ДАНИХ

База даних до цієї курсової роботи буде реалізована за допомогою системи управління базою даних PostgreSQL. Такий вибір пов'язаний із перевагами, які має ця СУБД. Зокрема можна виокремити багатьох SQL-стандартів, що можуть бути кращими ніж в інших системах [1].

PostgreSQL підтримує наповнення новими можливостями та функціоналом до базової СУБД за рахунок системи розширень, що була додана при розробці. Ця система відома своєю надійністю та стабільністю, що робить її надійним вибором для критичних застосунків. Перевагою PostgreSQL є масштабованість, оскільки СУБД підтримує оперування з великими об'ємами даних [1].

Ця система керування базою даних охоплює широкий функціонал можливостей для оптимізації продуктивності виконуваних запитів. Оскільки запити це одна з основних одиниць процедурного функціоналу будь-якої бази даних, то ефективність за часом є важливою складовою вибору СУБД [1].

PostgreSQL підтримує об'ємний кластер безпекових заходів із метою захисту даних. СУБД надає можливості надання прав певним ролям та користувачам, що супроводжується автентифікацією, авторизацією, шифруванням та хешуванням [1].

Якщо говорити про переваги, що не були застосовані в ході виконання цієї курсової роботи, то зокрема це повна підтримка транзакцій, що гарантує цілісність і надійність даних. PostgreSQL підтримує роботу з широким діапазоном файлових систем, наприклад JSON і JSONB [1].

Отже, PostgreSQL є досить доцільним вибором реляційної бази даних, що дозволить розробити ефективну інфраструктуру бази даних для цієї курсової роботи. Можливості цієї СУБД забезпечать ефективний імпорт даних та вдалий менеджмент інформацією [1].

## 1. Створення таблиць засобами SQL

Передусім створимо базу даних, за допомогою наступного скрипту:

```
CREATE DATABASE examus;
```

Створення таблиць бази даних та обмежень до них відбувалося за допомогою SQL-скриптів, які підтримує СУБД PostgreSQL. У вищенаведених розділах перелічено дванадцять таблиць, що необхідно реалізувати відповідно до інфраструктури бази даних визначеної теми.

Першою таблицею, яка створюється у нашій базі даних є “Тип структурного підрозділу”, яка описана у розділах “Опис предметного середовища” та “Побудова ER-моделі”. Створення цієї таблиці забезпечується наступним скриптом, наведеним у лістингу 1.1:

```
CREATE TABLE TypeOfDepartment(  
  id BIGSERIAL PRIMARY KEY NOT NULL,  
  type VARCHAR(50) NOT NULL  
);
```

Лістинг 1.1 — Створення таблиці “Тип структурного підрозділу”

У лістингу 1.1 наведено код генератора BIGSERIAL, який забезпечує автоматичну інкрементацію первинного ключа при імпорті даних за допомогою команди INSERT з DML запитів або COPY для імпорту з файлу.

Структура таблиці “TypeOfDepartment” для зберігання даних про типи структурних підрозділів, що визначені в описі предметного середовища опишемо структурою з таблиці 1.1.

Таблиця 1.1 — Структура таблиці “TypeOfDepartment”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
type	VARCHAR	50		Опис типу підрозділу

Забезпечимо створення таблиці “Структурного підрозділу”, атрибути сутності якої визначені нами у описі предметного середовища. Створимо цю таблицю за допомогою SQL скрипта, що наведений у лістингу 1.2:

```
CREATE TABLE StructureDepartment(  
  id BIGSERIAL PRIMARY KEY NOT NULL,  
  type VARCHAR(50) NOT NULL  
);
```

```

id BIGSERIAL PRIMARY KEY NOT NULL,
name VARCHAR(100) NOT NULL,
phoneNumber VARCHAR(30) NOT NULL,
email VARCHAR(50) NOT NULL,
website VARCHAR(50) NOT NULL,
type BIGSERIAL NOT NULL REFERENCES TypeOfDepartment(id)
);

```

Лістинг 1.2 — Створення таблиці “Структурний підрозділ”

У створені таблиці можемо побачити ключове слово REFERENCES, що відповідає за утворення зовнішнього ключа на таблицю “Тип підрозділу”.

Покажемо створену структуру таблиці “StructureDepartment”, що визначена у описі предметного середовища, а також що взаємодіє з таблицею “TypeOfDepartment” у таблиці 1.2.

Таблиця 1.2 — Структура таблиці “TypeOfDepartment”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
name	VARCHAR	100		Назва підрозділу
phoneNumber	VARCHAR	30		Номер мобільного
email	VARCHAR	50		Електронна пошта
website	VARCHAR	50		Адреса веб-сайту
type	BIGSERIAL		FK	Тип підрозділу

Наступною таблицею, яку ми створимо за допомогою SQL запитів буде “Кафедра”, яка є частиною структурного підрозділу. Під час створення таблиці “Кафедра” врахуємо всі атрибути та зовнішні ключі, які ми описали. Створення таблиці на основі сутності описаної в концептуальній моделі покажемо шляхом відображення SQL запиту, що наведений у лістингу 1.3:

```

CREATE TABLE Chair(
id BIGSERIAL PRIMARY KEY NOT NULL,
name VARCHAR(50) NOT NULL,
phoneNumber VARCHAR(30) NOT NULL,
email VARCHAR(50) NOT NULL,
website VARCHAR(50) NOT NULL,

```

```

structureDepartment BIGSERIAL NOT NULL REFERENCES
StructureDepartment(id)
);

```

Лістинг 1.3 — Створення таблиці “Кафедра”

Структуру таблиці “Chair” відобразимо за допомогою таблиці. У таблиці буде показана будова “Chair” створена SQL скриптом (таблиця 1.3).

Таблиця 1.3 — Структура таблиці “Chair”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
name	VARCHAR	50		Назва кафедри
phoneNumber	VARCHAR	30		Номер мобільного
email	VARCHAR	50		Електронна пошта
website	VARCHAR	50		Адреса веб-сайту
structureDepartment	BIGSERIAL		FK	Структурний підрозділ

Для реалізації сутності “Викладач” у базі даних для початку потрібно створити таблицю “Вчене звання”, на яке сутність “Викладач” посилається. Створення таблиці “Вчене звання” наведено у лістингу 1.4:

```

CREATE TABLE EducationalDegree(
id BIGSERIAL PRIMARY KEY NOT NULL,
educationalDegree VARCHAR(30) NOT NULL
);

```

Лістинг 1.4 — Створення таблиці “EducationalDegree”

Для таблиці “EducationalDegree”, створеної SQL скриптом покажемо її структурну таблицю. У таблиці 1.4 показано структуру “EducationalDegree” зі всіма атрибутами та ключами:



Таблиця 1.4 — Структура таблиці “EducationalDegree”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
educationalDegree	VARCHAR	30		Значення вч. звання

Створимо таблицю сутності “Вчитель” за атрибутами та зовнішніми ключами, що визначені у розділі “Опис предметного середовища” та “Побудова ER-моделі”. Використаємо SQL-запит, що наведений у лістингу 1.5:

```
CREATE TABLE Teacher(
  id BIGSERIAL PRIMARY KEY NOT NULL,
  firstName VARCHAR(50) NOT NULL,
  lastName VARCHAR(50) NOT NULL,
  patronymic VARCHAR(50) NOT NULL,
  beginningWorking DATE NOT NULL,
  chair BIGSERIAL NOT NULL REFERENCES Chair(id),
  educationalDegree BIGSERIAL NOT NULL REFERENCES EducationalDegree(id)
);
```

Лістинг 1.5 — Створення таблиці “Teacher”

Відобразимо структуру “Teacher” з її атрибутами та зовнішніми ключами, а також типами даних та правилами у таблиці 1.5:

Таблиця 1.5 — Структура таблиці “Teacher”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
firstName	VARCHAR	50		Ім'я
secondName	VARCHAR	50		Прізвище
patronymic	VARCHAR	50		По-батькові
beginningWorking	DATE			Дата влаштування
chair	BIGSERIAL		FK	Кафедра
educationalDegree	BIGSERIAL		FK	Вчене звання

Предметна область бази даних передбачає створення таблиці академічної групи студентів, яку ми визначили як сутність “Група” у вищезгаданих розділах. Цю таблицю ми створимо за допомогою SQL скрипта, який ми наведемо у лістингу 1.6. Після цього утворимо відповідну таблицю, що опише структуру імплементованої сутності.

```
CREATE TABLE AcademicGroup(
  id BIGSERIAL PRIMARY KEY NOT NULL,
  name VARCHAR(10) NOT NULL,
  chair BIGSERIAL NOT NULL REFERENCES Chair(id),
  academicCurator BIGSERIAL NOT NULL REFERENCES Teacher(id)
);
```

Лістинг 1.6 — Створення таблиці “AcademicGroup”

Тепер відобразимо структуру створеної таблиці нижче (таблиця 1.6) з відповідними атрибутами, первинним та зовнішніми ключами.

Таблиця 1.6 — Структура таблиці “AcademicGroup”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
name	VARCHAR	10		Назва групи
chair	BIGSERIAL		FK	Кафедра
academicCurator	BIGSERIAL		FK	Академічний куратор

Перед тим як створити таблицю сутності “Контроль” слід створити таблиці, на які “Контроль” посилатиметься. Зокрема предметною областю передбачено розробку таблиці “Тип контролю”. Створимо цю таблицю за допомогою скрипта, що наведений у лістингу 1.7.

```
CREATE TABLE TypeOfControl(
  id BIGSERIAL NOT NULL PRIMARY KEY,
  typeOfControl VARCHAR(50) NOT NULL,
  hours INTEGER NOT NULL
);
```

Лістинг 1.7 — Створення таблиці “TypeOfControl”

Наведемо структуру створеної сутності у таблиці 1.7, де відобразимо всі атрибути з їх типами даних, первинний та зовнішній ключі, а також призначення цих атрибутів.

Таблиця 1.7 — Структура таблиці “TypeOfControl”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
typeOfControl	VARCHAR	50		Опис типу контролю
hours	INTEGER			Кількість годин відведених на контроль

Опис предметного середовища передбачує створення таблиці “Оцінка”, яка виставляється студенту по результатам проведення його контролю. У таблиці маємо реалізувати поля, що відповідають за кількість балів, національну оцінку та оцінку в шкалі ЄКТС. Створення таблиці “Оцінка” відображено у лістингу 1.8:

```
CREATE TABLE Mark(
  id BIGSERIAL NOT NULL PRIMARY KEY,
  score NUMERIC NOT NULL,
  nationalMark VARCHAR(15) NOT NULL,
  markECTS VARCHAR(5) NOT NULL
);
```

Лістинг 1.8 — Створення таблиці “Mark”

Відобразимо структуру імплементованої сутності у вигляді таблиці разом із первинним та зовнішніми ключами, атрибутами та їх типами даних. Наглядно продемонструємо це у таблиці 1.8.

Таблиця 1.8 — Структура таблиці “Mark”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
score	NUMERIC			Кількість балів

Продовження таблиці 1.8

Ім'я поля	Тип даних	Розмір	Ключ	Опис
nationalMark	VARCHAR	15		Національна оцінка
markECTS	VARCHAR	5		Оцінка ECTS

Кожний предмет має власний тип, тому виникає необхідність створення таблиці “Типу предмету”. Це твердження передбачене описом предметного середовища та ER-моделлю бази даних. Створення таблиці “Тип предмету” відображено у лістингу 1.9.

```
CREATE TABLE TypeOfSubject(
  id BIGSERIAL NOT NULL PRIMARY KEY,
  type VARCHAR(20) NOT NULL
);
```

Лістинг 1.9 — Створення таблиці “TypeOfSubject”

У таблиці 1.9 показано структуру сутності “TypeOfSubject” з її атрибутами, ключами та типами даних.

Таблиця 1.9. — Структура таблиці “TypeOfSubject”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
type	VARCHAR	20		Опис типу предмету

Відповідно перейдемо до створення таблиці “Предмет”, яка міститиме інформацію про дисципліни, які викладаються в університеті. Згідно з описом предметного середовища предмет має унікальний ідентифікатор, назву, тип предмету та кількість годин, яка визначена для цього предмету. Створення таблиці “Предмет” наведено у лістингу 1.10:

```
CREATE TABLE Subject(
  id BIGSERIAL NOT NULL PRIMARY KEY,
  name VARCHAR(50) NOT NULL,
  hours INTEGER NOT NULL,
  typeOfSubject BIGSERIAL NOT NULL REFERENCES TypeOfSubject(id)
);
```

Лістинг 1.10 — Створення таблиці “Subject”

Відповідно до дій вище, необхідно створити структуру створеної таблиці. Будова таблиці “Subject” описана у таблиці 1.10.

Таблиця 1.10 — Структура таблиці “Subject”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
name	VARCHAR	50		Назва предмету
hours	INTEGER			Кількість годин
typeOfSubject	BIGSERIAL		FK	Тип предмету

Також створимо таблицю “Студент”, яка вимагається концептуальною моделлю бази даних. Здобувач освіти має бути присутнім у оцінюванні його знань під час будь-якого типу контролю. Створення таблиці “Студент” описується лістингом 1.11.

```
CREATE TABLE Student(
  id BIGSERIAL NOT NULL PRIMARY KEY,
  firstName VARCHAR(50) NOT NULL,
  secondName VARCHAR(50) NOT NULL,
  patronymic VARCHAR(50) NOT NULL,
  dateOfEntry DATE NOT NULL,
  academicGroup BIGSERIAL NOT NULL REFERENCES AcademicGroup(id)
);
```

Лістинг 1.11 — Створення таблиці “Student”

Відобразимо структуру таблиці “Student” за допомогою моделі, яку ми застосовували вище. Покажемо будову сутності “Student” у таблиці 1.11.

Таблиця 1.11 — Структура таблиці “Student”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
firstName	VARCHAR	50		Ім'я
secondName	VARCHAR	50		Прізвище
patronymic	VARCHAR	50		По-батькові
dateOfEntry	DATE			Дата вступу
academicGroup	BIGSERIAL		FK	Академічна група

Створимо таблицю проведення контролю, врахувавши що у предметній області вказано про проведення будь-якого типу контролю для конкретного студента з конкретним викладачем в окрему дату з точною оцінкою або атестацією з визначеного предмету.

Створення таблиці “Контроль” можемо побачити у лістингу 1.12.

```
CREATE TABLE "Control"(
  id BIGSERIAL NOT NULL PRIMARY KEY,
  attestation BOOL,
  dateOfControl DATE NOT NULL,
  typeOfControl BIGSERIAL NOT NULL REFERENCES TypeOfControl(id),
  mark INTEGER REFERENCES Mark(id),
  subject BIGSERIAL NOT NULL REFERENCES Subject(id),
  student BIGSERIAL NOT NULL REFERENCES Student(id),
  teacher BIGSERIAL NOT NULL REFERENCES Teacher(id)
);
```

Лістинг 1.12 Створення таблиці “Control”

Структура таблиці “Control” відповідно до дій вище має бути показана в табличному записі. Покажемо всі атрибути, зовнішні та первинний ключ сутності у таблиці 1.12.

Таблиця 1.12 — Структура таблиці “Control”

Ім'я поля	Тип даних	Розмір	Ключ	Опис
id	BIGSERIAL		PK	Унікальний ідентифікатор
attestation	BOOL			Атестація
dateOfControl	DATE			Дата контролю
typeOfControl	BIGSERIAL		FK	Тип контролю
mark	INTEGER		FK	Оцінка
subject	BIGSERIAL		FK	Предмет
student	BIGSERIAL		FK	Студент
teacher	BIGSERIAL		FK	Викладач

## 2. Створення обмежень бази даних засобами SQL

Створення обмежень для таблиць бази даних є важливою частиною розробки інфраструктури інформаційної моделі. За допомогою обмежень ми можемо уникнути порушення цілісності та невалідності даних. У цій курсовій роботі база даних також забезпечена обмеженнями

```
ALTER TABLE TypeOfDepartment ADD CONSTRAINT uniqueDepartmentType
UNIQUE(type);
```

### Лістинг 2.1 — Обмеження унікальності для типу підрозділу

У лістингу 2.1 показано створення обмеження для таблиці типів структурних підрозділів з ключовим словом UNIQUE. Кожний тип підрозділу має бути унікальним для уникнення дублювання і збереження надлишкових даних.

```
ALTER TABLE StructureDepartment ADD CONSTRAINT uniqueDepartmentName
UNIQUE(name);
```

### Лістинг 2.2 — Обмеження унікальності імені для структурного підрозділу

У лістингу 2.2. показано створення обмеження для назв структурних підрозділів. Кожний структурний підрозділ має мати власне унікальне ім'я відмінне від інших.

```
ALTER      TABLE      StructureDepartment      ADD      CONSTRAINT
uniqueDepartmentPhoneNumber UNIQUE(phoneNumber);
```

#### Лістинг 2.3 — Обмеження унікальності мобільного номеру структурного підрозділу

У лістингу 2.3 наведено створення обмеження для унікальності номеру мобільного телефону структурного підрозділу, оскільки підрозділи не можуть мати спільні контакти. Такі ж обмеження стосуються адрес електронних пошт та веб-сайтів, що відображено у лістингу 2.4.

```
ALTER      TABLE      StructureDepartment      ADD      CONSTRAINT
uniqueDepartmentEmail UNIQUE(email);
```

```
ALTER      TABLE      StructureDepartment      ADD      CONSTRAINT
uniqueDepartmentWebsite UNIQUE(website);
```

#### Лістинг 2.4 — Обмеження унікальності контактних даних структурних підрозділів

Для таблиці “Кафедра” виникає необхідність створити відповідні обмеження, оскільки ця таблиця наділена атрибутами назви, мобільного номеру, електронної пошти та веб-ресурсу. Створення цих обмежень наведемо у лістингу 2.5.

```
ALTER TABLE Chair ADD CONSTRAINT ChairUniqueName UNIQUE(name);
ALTER TABLE Chair ADD CONSTRAINT ChairUniquePhoneNumber
UNIQUE(phoneNumber);
ALTER TABLE Chair ADD CONSTRAINT ChairUniqueWebsite UNIQUE(website);
ALTER TABLE Chair ADD CONSTRAINT ChairUniqueEmail UNIQUE(email);
```

#### Лістинг 2.5 — Створення обмежень унікальності для персональних даних кафедри

Певні обмеження також мають стосуватися академічних груп. Кожна група має власне ім'я та власного академічного куратора. При створенні таблиці “AcademicGroup” (лістинг 1.6) було вказано, що ми створюємо



зовнішній ключ на таблицю “Teacher”. Така структура забезпечує створення зв’язку “один до багатьох”, проте не “один до одного” як вказано у описі предметного середовища. Тому маємо створити обмеження унікальності для атрибуту “academicCurator”, щоб забезпечити зв’язок “один до одного”. Таке обмеження описане у лістингу 2.6.

```
ALTER TABLE AcademicGroup ADD CONSTRAINT GroupUniqueCurator
UNIQUE(academicCurator);
```

Лістинг 2.6 — Обмеження унікальності академічного куратора групи

Кожна група має мати унікальне ім’я, щоб уникнути дублювань та невідповідностей у різних типах даних. Тому за допомогою ключового слова UNIQUE потрібно обмежити атрибут таблиці “AcademicGroup”, що має символічний тип “name”. Таке обмеження показано у лістингу 2.7.

```
ALTER TABLE AcademicGroup ADD CONSTRAINT GroupUniqueName
UNIQUE(name);
```

Лістинг 2.7 — Створення обмеження унікальності імені групи

Аналогічно до типів структурних підрозділів необхідно створити обмеження для унікальності вченого звання викладачів, що містяться у таблиці “EducationalDegree”. Створення такого обмеження наведено у лістингу 2.8.

```
ALTER TABLE EducationalDegree ADD CONSTRAINT DegreeUniqueValue
UNIQUE(educationalDegree);
```

Лістинг 2.8 — Обмеження унікальності вченого звання

Подібні обмеження слід створити для типу предмету, назви предмету та типу контролю, які проводяться в університеті. Для цього використаємо скрипт, який наведений у лістингу 2.9.

```
ALTER TABLE TypeOfSubject ADD CONSTRAINT SubjectUniqueType
UNIQUE(type);
```

```
ALTER TABLE TypeOfControl ADD CONSTRAINT ControlUniqueType
UNIQUE(typeOfControl);
```

```
ALTER TABLE Subject ADD CONSTRAINT SubjectUniqueName UNIQUE(name);
```

Лістинг 2.9 — Обмеження для типу предмету, назви предмету та типу контролю

### 3. Створення користувачів бази даних

Згідно з аналізом предметного середовища необхідно розробити користувачів та їх ролі у базі даних. Описом передбачено трьох користувачів, які будуть використовувати сховище інформації, зокрема: здобувач освіти, викладач, працівник кафедри та працівник університету. Ці користувачі мають мати різні права на взаємодію із базою.

Працівники університету та кафедр мають право на перегляд поточних результатів контролю, тому для такого виду користувачів необхідно створити окрему роль, яка дозволяє їм переглядати всі необхідні дані з таблиці “Control” та з нею пов’язані. Доречним буде надання прав таким працівникам на внесення змін до певних даних. Відповідний код створення ролей та надання їм прав показано у лістингу 3.1.

```
CREATE ROLE controlReviewerRole;
GRANT SELECT ON "Control", Teacher,
Student, TypeOfControl, TypeOfSubject TO controlReviewerRole;

CREATE ROLE controlEditorRole;
GRANT INSERT, SELECT, UPDATE, DELETE ON "Control" TO
controlEditorRole;
```

Лістинг 3.1 — Створення ролей працівників університету та кафедр і надання їм прав

Також необхідно проводити моніторинг та менеджмент інформації про персональні дані та успішність студентів, що здобувають освіту в університеті. Відповідне твердження можемо сказати і про дисципліни, які викладаються на кафедрах. Відповідне створення ролей і надання їм прав показано у лістингу 3.2.

```
CREATE ROLE studentManagement;
GRANT INSERT, DELETE, UPDATE, SELECT ON Student, AcademicGroup TO
studentManagement;

CREATE ROLE subjectManagement;
GRANT INSERT, DELETE, UPDATE, SELECT ON Subject TO
subjectManagement;
```

Лістинг 3.2 — Створення ролей моніторингу студентів і предметів та наділення їх правами

Згідно з аналізом предметної області створимо визначених зарання користувачів і надамо їм права на оперування певними даними. Такими користувачами будуть “Студент”, “Кафедра”, “Викладач” та “Працівник університету”. Створення користувачів та надання їм прав визначених вище ролей наведено у лістингу 3.3.

```
CREATE USER student WITH PASSWORD 'abc';
CREATE USER cathedra WITH PASSWORD 'abc';
CREATE USER university WITH PASSWORD 'abc';
CREATE USER teacher WITH PASSWORD 'abc';

GRANT controlReviewerRole TO student;

GRANT controlReviewerRole TO Teacher;
GRANT controlEditorRole TO Teacher;

GRANT studentManagement TO cathedra;
GRANT subjectManagement TO cathedra;

GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO university;
```

Лістинг 3.3 — Створення користувачів і наділення їх правами

#### 4. ER-діаграма створена засобами СУБД

Відповідно до вимог курсової роботи необхідно відобразити ER-діаграму згенеровану засобами системи управління базою даних. Можливості PostgreSQL дозволяють згенерувати таку діаграму. Схематичне зображення бази даних відображено на рисунку 1.

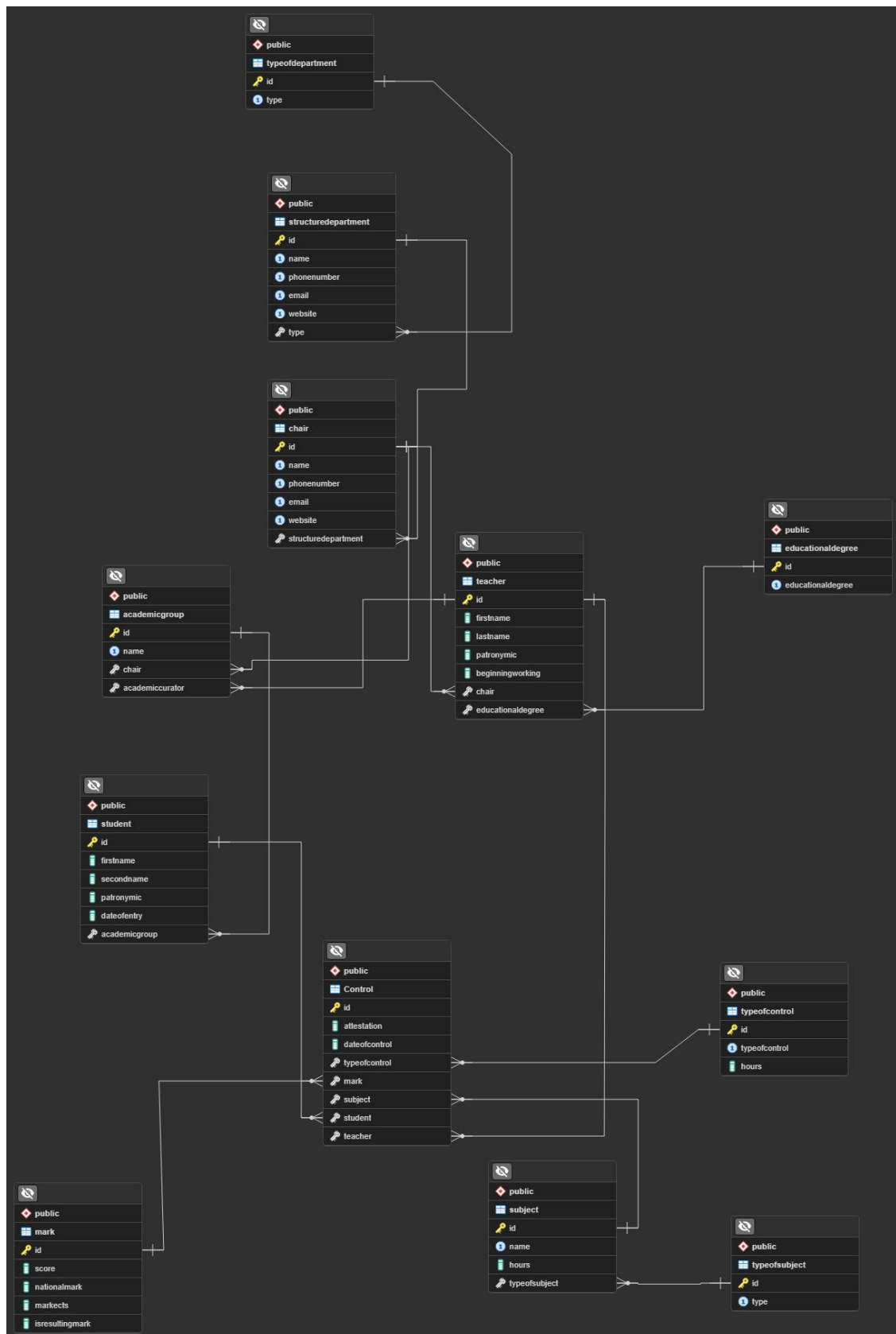


Рисунок 1 — Діаграма відношень між сутностями згенерована засобами СУБД

## 7 РОБОТА З БАЗОЮ ДАНИХ

У цьому розділі наведемо усі наявні скрипти відповідно до вимог, що наведені для цієї курсової роботи.

### 1. Генератори

У ході виконання курсової роботи та створення фізичної бази даних в якості генераторів було використано тип даних BIGSERIAL, який автоматично інкрементує значення. Цей тип даних використовувався при створенні таблиць, скрипти формування яких наведено у розділі “Реалізація бази даних”.

### 2. Збережені процедури та функції

Сумарно в курсовій роботі було створено десять збережених функцій та процедур. Розроблений процедурно-орієнтований функціонал дозволяє побачити потрібний набір інформації без використання додаткових складних та об’ємних SQL запитів [7].

У лістингу 2.1 наведено збережену процедуру, функціонал якої змінює кафедру на якій працює викладач. На вхід процедура приймає ID викладача та ID кафедри, яка буде для нього встановлена. У програмному коді збереженої процедури передбачена валідація вхідних даних. У випадку некоректних даних процедура повідомить користувача про помилку. Якщо вхідні дані валідні, то для викладача із заданим унікальним ідентифікатором оновляться дані про кафедру.

```
CREATE OR REPLACE PROCEDURE changeChairOfTeacher(teacherID INTEGER,
newChairID INTEGER)
LANGUAGE plpgsql AS $$
DECLARE
    oldChair INTEGER;
BEGIN
    IF EXISTS(SELECT 1 FROM Teacher WHERE Teacher.id = teacherID)
AND
    EXISTS(SELECT 1 FROM Chair WHERE Chair.id = newChairID) THEN
```

```

SELECT chair INTO oldChair FROM Teacher WHERE Teacher.id
= teacherID;

UPDATE Teacher SET Chair = newChairID WHERE Teacher.id =
teacherID;

RAISE NOTICE 'Teacher with id % has change chair from %
to %', teacherID, oldChair, newChairID;
ELSE
    RAISE NOTICE 'Chair or teacher is incorrect';
END IF;
END;
$$;

```

#### Лістинг 2.1 — Збережена процедура оновлення кафедри викладача

У лістингу 2.2 наведено функцію, у програмному коді якої відбувається вибірка даних. Суть функції полягає у відображенні таблиці штатних працівників певного структурного підрозділу. Зокрема мова йде про викладачів, які працюють у певному інституті або факультеті. Вибірка даних відбувається за принципом вибору повного імені працівника, кафедри і структурного підрозділу, де ID структурного підрозділу рівний вхідному параметру, який на вхід приймає функція.

```

CREATE OR REPLACE PROCEDURE changeChairOfTeacher(teacherID INTEGER,
newChairID INTEGER)
LANGUAGE plpgsql AS $$
DECLARE
    oldChair INTEGER;
BEGIN
    IF EXISTS(SELECT 1 FROM Teacher WHERE Teacher.id = teacherID)
AND
    EXISTS(SELECT 1 FROM Chair WHERE Chair.id = newChairID) THEN
        SELECT chair INTO oldChair FROM Teacher WHERE Teacher.id
= teacherID;

        UPDATE Teacher SET Chair = newChairID WHERE Teacher.id =
teacherID;

```

```

        RAISE NOTICE 'Teacher with id % has change chair from %
to %', teacherID, oldChair, newChairID;
    ELSE
        RAISE NOTICE 'Chair or teacher is incorrect';
    END IF;
END;
$$;

```

## Лістинг 2.2 — Функція відображення штатних працівників підрозділу

Функціонал бази даних також забезпечений процедурою, яка оновлює персональні дані студента. На вхід процедура приймає нові прізвище, ім'я і по-батькові, а також унікальний ідентифікатор студента. Програмний код забезпечений перевіркою валідації даних за допомогою предикату EXISTS, який повертає булеве значення TRUE або FALSE. Якщо студент із заданим ID існує, то його дані будуть оновлені, інакше процедура надасть повідомлення про відсутність особи із наданим вхідним параметром. Код до цієї збереженої процедури відображений у лістингу 2.3.

```

CREATE OR REPLACE PROCEDURE updateStudentPersonalData(studentID
INTEGER, newFirstName VARCHAR(50), newSecondName VARCHAR(50),
newPatronymic VARCHAR(50))
LANGUAGE plpgsql AS $$
BEGIN
    IF EXISTS(SELECT 1 FROM Student WHERE Student.id = studentID)
THEN
        UPDATE Student SET firstname = newFirstName,
        secondname = newSecondName, patronymic = newPatronymic
        WHERE id = studentID;
        RAISE NOTICE 'Name of student with id % changed
to % % %', studentID, newSecondName, newFirstName, newPatronymic;

    ELSE
        RAISE NOTICE 'There is no student with such id';
    END IF;

```

END;

\$\$;

### Лістинг 2.3 — Процедура оновлення персональних даних студента

Наступна функція відповідає за підрахунок і повертає цілочисельне значення у результаті свого виконання. Її головна ідея полягає у підрахунку студентів, які незадовільно склали семестровий контроль поточного року. Дані про поточний рік беруться із константи `CURRENT_DATE`, що передбачена функціоналом СУБД. Проводиться вибірка із використанням функції `COUNT`, що підраховує кількість. У ході виконання функції проводиться вибірка, де підраховується кількість студентів, де тип контролю — це семестровий контроль, дата контролю рівна поточному року, а значення атестації негативне. У лістингу 2.4 наведено програмний код функції.

```
CREATE OR REPLACE FUNCTION defineUnsuccessfulStudents(department
INTEGER)
    RETURNS INTEGER AS $$

    DECLARE
        resultValue INTEGER;
    BEGIN
        SELECT COUNT(Student.id) INTO resultValue
        FROM StructureDepartment
        JOIN Chair ON Chair.structureDepartment =
StructureDepartment.id
        JOIN AcademicGroup ON AcademicGroup.chair = Chair.id
        JOIN Student ON Student.academicgroup = AcademicGroup.id
        JOIN "Control" ON Student.id = "Control".student
        WHERE StructureDepartment = department
        AND "Control".typeOfControl = (SELECT id FROM TypeOfControl
WHERE typeOfControl = 'Semester control')

        AND EXTRACT(YEAR FROM CURRENT_DATE) = EXTRACT(YEAR FROM
"Control".dateOfControl)
```



```
AND "Control".attestation = FALSE;
```

```
RETURN resultValue;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

Лістинг 2.4 — Функція обрахунку студентів, що незадовільно склали семестровий контроль

У лістингу 2.5 відображено функцію, яка відображає наповнення штату певного структурного підрозділу за пороговим значенням стажу роботи та вченим званням працівника. На вхід функція приймає параметри унікальних ідентифікаторів структурного підрозділу та вченого звання, а також порогове значення досвіду роботи. У результаті своєї роботи функція повертає таблицю заданої структури. Програмний код забезпечений вибіркою даних, де відбирається ім'я викладача, структурний підрозділ, його вчене звання та досвід роботи, де перелічені параметри відповідають вхідним параметрам функції.

```
CREATE OR REPLACE FUNCTION queryDefinedStaffDepartment(department
INTEGER, experineceValue INTEGER, degreeID INTEGER)
RETURNS TABLE("Department" VARCHAR(150), "Teacher" TEXT, "Degree"
VARCHAR(100), "Experience" NUMERIC) AS $$
BEGIN
    RETURN QUERY SELECT StructureDepartment.name AS "Department",
        Teacher.lastname || ' ' || Teacher.firstname || ' ' ||
Teacher.patronymic AS "Teacher",
        EducationalDegree.educationaldegree AS "Degree",
        EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningworking) AS "Experience"
    FROM StructureDepartment
    JOIN Chair ON Chair.structuredepartment =
StructureDepartment.id
    JOIN Teacher ON Teacher.chair = Chair.id
```

```

        JOIN EducationalDegree ON EducationalDegree.id =
Teacher.educationalDegree
        WHERE StructureDepartment.id = department
        AND EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningworking) >= experineceValue
        AND Teacher.educationalDegree = degreeID;
    END;
    $$ LANGUAGE plpgsql;

```

Лістинг 2.5 — Функція відображення штатного складу підрозділу за певними параметрами

Функціонал бази даних передбачає наявність процедури, яка виставляє студенту оцінку по певному виду контрольного заходу. На вхід процедура приймає унікальний ідентифікатор студента, дату заходу, ідентифікатор предмету, ідентифікатор викладача, оцінку в шкалі “0-100” та ідентифікатор контролю. Програмний код процедури забезпечений валідацією вхідних даних. Програмний код відображено у лістингу 2.6.

```

CREATE OR REPLACE PROCEDURE createMarkForStudent(
    studentID INTEGER,
    requiredDate DATE,
    subjectID INTEGER,
    teacherID INTEGER,
    markScore INTEGER,
    controlID INTEGER
)
LANGUAGE plpgsql AS $$
DECLARE
    markID INTEGER;
BEGIN
    IF controlID != (SELECT id FROM TypeOfControl WHERE
typeofcontrol = 'Semester control')
        AND controlID != (SELECT id FROM TypeOfControl WHERE
typeofcontrol = 'Calendar control')
    THEN

```

```

        IF NOT EXISTS(SELECT 1 FROM Student WHERE Student.id =
studentID) THEN
            RAISE NOTICE 'There is no such student in table.';
            RETURN;
        ELSIF NOT EXISTS(SELECT 1 FROM Subject WHERE Subject.id
= subjectID) THEN
            RAISE NOTICE 'There is no such subject in table.';
            RETURN;
        ELSIF NOT EXISTS(SELECT 1 FROM Teacher WHERE Teacher.id
= teacherID) THEN
            RAISE NOTICE 'There is no such teacher in table.';
            RETURN;
        ELSIF NOT EXISTS(SELECT 1 FROM TypeOfControl WHERE
TypeOfControl.id = controlID) THEN
            RAISE NOTICE 'There is no such type of control in
table.';

            RETURN;
        ELSIF markScore < 0 OR markScore > 100 THEN
            RAISE NOTICE 'Mark is not in required diapason.';
        ELSE
            SELECT id INTO markID FROM Mark WHERE Mark.score =
markScore AND Mark.isResultingMark = FALSE;
            INSERT INTO "Control"(student, dateofcontrol,
subject, teacher, mark, typeofcontrol) VALUES
            (studentID, requiredDate, subjectID, teacherID,
markID, controlID);
            RAISE NOTICE 'New data about student control is
created.';

            RAISE NOTICE 'Control id: %',
            (SELECT id FROM "Control" WHERE student = studentID
AND teacher = teacherID
            AND subject = subjectID AND mark = markID AND
typeofcontrol = controlID AND
            dateofcontrol = requiredDate);

```

```

        RAISE NOTICE 'Student: %',
        (SELECT Student.secondname || ' ' ||
Student.firstname || ' ' || Student.patronymic FROM Student
        WHERE Student.id = studentID);
        RAISE NOTICE 'Teacher: %',
        (SELECT Teacher.lastname || ' ' ||
Teacher.firstname || ' ' || Teacher.patronymic FROM Teacher WHERE
        Teacher.id = teacherID);
        RAISE NOTICE 'Subject: %', (SELECT Subject.name
FROM Subject WHERE Subject.id = subjectID);
        RAISE NOTICE 'Event: %', (SELECT
TypeOfControl.typeofcontrol FROM TypeOfControl WHERE TypeOfControl.id =
controlID);

        RAISE NOTICE 'Mark: %', markScore;
    END IF;
ELSE
        RAISE NOTICE 'You can not set numeric marks for calendar
and semester control';
    END IF;
END;
$$;

```

#### Лістинг 2.6 — Процедура виставлення оцінки студенту

Було створено окрему процедуру, яка відображає залікову книжку студента. Функціонал забезпечений поступовим виведенням на екран відомостей про те, як студент отримував підсумкові оцінки з певного предмету. Програмний код процедури забезпечений наявністю курсора та поступовим виведенням даних із запиту у циклі. Повний склад процедури відображено у лістингу 2.7.

```

CREATE OR REPLACE PROCEDURE studentRecordBook(studentID INTEGER)
LANGUAGE plpgsql AS $$
DECLARE
    recordBookCursor CURSOR FOR
    SELECT Subject.name AS "Subject",

```

```

        "Control".dateofcontrol AS "Date",
        Mark.score AS "Total score",
        Mark.marECTS AS "ECTS"
    FROM "Control"
    JOIN Subject ON Subject.id = "Control".subject
    JOIN Mark ON "Control".mark = Mark.id
    WHERE "Control".student = studentID
    AND "Control".typeofcontrol = (SELECT id FROM TypeOfControl
    WHERE typeofcontrol = 'Exam');

    recordBookRecord RECORD;
BEGIN
    OPEN recordBookCursor;

    RAISE NOTICE 'Record book of %',
        (SELECT Student.secondname || ' ' || Student.firstname || ' '
        || Student.patronymic
        FROM Student WHERE Student.id = studentID);
    LOOP
        FETCH recordBookCursor INTO recordBookRecord;
        EXIT WHEN NOT FOUND;
        RAISE NOTICE 'Subject: % | Date: % | Score: % | ECTS: %
        ',
            recordBookRecord."Subject", recordBookRecord."Date",
            recordBookRecord."Total score", recordBookRecord."ECTS";
    END LOOP;

    CLOSE recordBookCursor;

END;
$$;

```

Лістинг 2.7. — Процедура відображення залікової книжки студента

Наступна процедура відображає дані про студента за вхідними параметрами про пошук. Процедура на вхід приймає прізвище, ім'я і по-

батькові студента. Якщо студент існує в базі даних, то відображається інформація про те, у якій групі він вчиться, в складі якої кафедри та структурного підрозділу. Програмний код процедури забезпечений валідацією даних. У лістингу 2.8 наведений програмний код процедури.

```
CREATE OR REPLACE PROCEDURE findStudent(f_secondName VARCHAR(50),
f_firstName VARCHAR(50), f_patronymic VARCHAR(50))
LANGUAGE plpgsql AS $$

DECLARE
    studentRecord RECORD;

BEGIN
    IF EXISTS(SELECT 1 FROM Student WHERE firstname = f_firstname
              AND secondname = f_secondname
              AND patronymic = f_patronymic) THEN
        SELECT Student.secondName || ' ' ||
Student.firstName || ' ' || Student.patronymic AS "Student",
              AcademicGroup.name AS "Group", Chair.name AS
"Chair", StructureDepartment.name AS "Department"
              INTO studentRecord
              FROM Student
              JOIN AcademicGroup ON AcademicGroup.id =
Student.academicgroup
              JOIN Chair ON Chair.id = AcademicGroup.chair
              JOIN StructureDepartment ON
StructureDepartment.id = Chair.structuredepartment
              WHERE firstname = f_firstname
              AND secondname = f_secondname
              AND patronymic = f_patronymic;
        RAISE NOTICE 'Student: %',
studentRecord."Student";
        RAISE NOTICE 'Group: %', studentRecord."Group";
        RAISE NOTICE 'Chair: %', studentRecord."Chair";
```

```

        RAISE NOTICE 'Structure Department: %',
studentRecord."Department";
    ELSE
        RAISE NOTICE 'There is no such student in University.';
    END IF;
END;
$$;

```

#### Лістинг 2.8 — Пошук студента за його персональними даними

У лістингу 2.9 наведено функцію, яка підраховує в поточному році кількість студентів, що незадовільно склали екзамен та підлягають відрахуванню. Програмний код функції забезпечений оголошенням змінної та вибіркою даних. Вибірка даних імпортує у змінну число та повертає його у результаті виконання функції.

```

CREATE OR REPLACE FUNCTION countDismissalStudents()
RETURNS INTEGER AS $$
DECLARE
    resultVariable INTEGER;
BEGIN
    SELECT COUNT(DISTINCT Student.secondname || ' ' ||
Student.firstname || ' ' || Student.patronymic)
    INTO resultVariable
    FROM "Control"
    JOIN Student ON Student.id = "Control".student
    JOIN Mark ON Mark.id = "Control".mark
    WHERE "Control".typeOfControl = (SELECT id FROM TypeOfControl
WHERE typeofcontrol = 'Exam')
    AND Mark.score < 60
    AND EXTRACT(YEAR FROM CURRENT_DATE) = EXTRACT(YEAR FROM
"Control".dateofcontrol);
    RETURN resultVariable;
END;
$$ LANGUAGE plpgsql;

```

#### Лістинг 2.9 — Функція підрахунку студентів, що підлягають відрахуванню

Остання процедура має функціонал відображення необхідної інформації про університет. Тобто кількість кожного типу його структурних підрозділів, контактні дані його структурних підрозділів. Програмний код забезпечений курсором та циклом, який використовує курсор для відображення даних. Код до цієї функції відображений у лістингу 2.10.

```
CREATE OR REPLACE PROCEDURE universityInfo()
LANGUAGE plpgsql AS $$
DECLARE
    amountOfFaculties INTEGER;
    amountOfInsistutes INTEGER;
    amountOfColleges INTEGER;
    amountOfProjInst INTEGER;
    departmentCursor CURSOR FOR
    SELECT StructureDepartment.name AS "Department",
    StructureDepartment.phoneNumber AS "Phone Number",
    StructureDepartment.website AS "Website",
    StructureDepartment.email AS "Email",
    TypeOfDepartment.type AS "Type"
    FROM StructureDepartment
    JOIN TypeOfDepartment ON TypeOfDepartment.id =
StructureDepartment.type;
    departmentRecord RECORD;
BEGIN
    SELECT COUNT(*) INTO amountOfFaculties FROM
StructureDepartment
    WHERE type = (SELECT id FROM TypeOfDepartment WHERE type =
'Faculty');
    SELECT COUNT(*) INTO amountOfInsistutes FROM
StructureDepartment
    WHERE type = (SELECT id FROM TypeOfDepartment WHERE type =
'Research and Educational Institute');
    SELECT COUNT(*) INTO amountOfColleges FROM StructureDepartment
```



```

WHERE type = (SELECT id FROM TypeOfDepartment WHERE type =
'Vocational College');
SELECT COUNT(*) INTO amountOfProjInst FROM StructureDepartment
WHERE type = (SELECT id FROM TypeOfDepartment WHERE type =
'Projecting Institute');

RAISE NOTICE 'Information about University';
RAISE NOTICE 'Faculties: %', amountOfFaculties;
RAISE NOTICE 'Insitutes: %', amountOfInsistutes;
RAISE NOTICE 'Colleges: %', amountOfColleges;
RAISE NOTICE 'Projectin institues: %', amountOfProjInst;

OPEN departmentCursor;

LOOP
    FETCH departmentCursor INTO departmentRecord;
    EXIT WHEN NOT FOUND;
    RAISE NOTICE '-----';
    RAISE NOTICE '%', departmentRecord."Department";
    RAISE NOTICE '%', departmentRecord."Phone Number";
    RAISE NOTICE '%', departmentRecord."Website";
    RAISE NOTICE '%', departmentRecord."Email";
    RAISE NOTICE '%', departmentRecord."Type";
END LOOP;

CLOSE departmentCursor;

END;
$$;

```

Лістинг 2.10 — Процедура відображення інформації про університет

### 3. Тригери

Тригери — це процедурне утворення, яке реагує на певні дії, при взаємодії із таблицею. Тригери використовуються при додаванні, видаленні або оновленні даних. У базі даних цієї курсової роботи також були застосовані тригери для покращення цілісності та ефективності заповнення інформації у базі [6].

У ході виконання курсової роботи було реалізовано сім тригерів різних за суттю. Програмний код та опис тригерних функцій та створення тригерів для таблиць наведено нижче.

У лістингу 3.1 наведено тригер, який валідує значення при вставці оцінки у таблицю. Оскільки кількість балів оцінки має задовільняти шкалу “0-100” і шкалу ЄКТС “A-F”, то виникає необхідність провести валідацію вхідних даних. Програмний код тригерної функції забезпечений валідацією кількості балів. Тобто якщо бал не входить у відрізок [0; 100], то тригерна функція оповістить користувача повідомленням із використанням виключної ситуації. В інакшому випадку підбирається відповідне значення національної оцінки та оцінки за ЄКТС.

```
CREATE OR REPLACE FUNCTION validateMarkTriggerFunction() RETURNS
TRIGGER AS $$
BEGIN
    IF NEW.score < 0 OR NEW.score > 100 THEN
        RAISE EXCEPTION 'Значення оцінки має входити у діапазон
від 0 до 100 балів.';
    END IF;

    IF NEW.isResultingMark = TRUE THEN
        NEW.nationalMark := CASE
            WHEN NEW.score BETWEEN 0 AND 39 THEN 'Not allowed'
            WHEN NEW.score BETWEEN 40 AND 59 THEN 'Not enough'
            WHEN NEW.score BETWEEN 60 AND 64 THEN 'Enough'
```

```

        WHEN NEW.score BETWEEN 65 AND 74 THEN
'Satisfactory'

        WHEN NEW.score BETWEEN 75 AND 84 THEN 'Good'
        WHEN NEW.score BETWEEN 85 AND 94 THEN 'Very Good'
        ELSE 'Excellent'
    END;

    NEW.markeCTS := CASE
        WHEN NEW.score BETWEEN 0 AND 39 THEN 'F'
        WHEN NEW.score BETWEEN 40 AND 59 THEN 'E'
        WHEN NEW.score BETWEEN 60 AND 64 THEN 'D'
        WHEN NEW.score BETWEEN 65 AND 74 THEN 'C'
        WHEN NEW.score BETWEEN 75 AND 84 THEN 'B'
        WHEN NEW.score BETWEEN 85 AND 94 THEN 'B+'
        ELSE 'A'
    END;
ELSE
    NEW.nationalMark := NULL;
    NEW.markeCTS := NULL;
END IF;

RETURN NEW;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

```

### Лістинг 3.1 — Тригерна функція валідації оцінки

Ця тригерна функція має бути прив’язана до відповідної для неї таблиці. Зокрема такою таблицею є “Mark”. Щоб створити тригер для цієї сутності необхідно виконати певний запит, де обов’язково потрібно вказати, що вхідні дані мають перевірятися після вставки. Код створення тригеру наведений у лістингу 3.2.

```

CREATE TRIGGER markValidationTrigger
BEFORE INSERT ON Mark
FOR EACH ROW
EXECUTE FUNCTION validateMarkTriggerFunction();

```

### Лістинг 3.2 — Створення тригера для таблиці “Mark”

Оскільки навчання відбувається в межах українського закладу вищої освіти, то виникає необхідність провести валідацію певних персональних даних. Зокрема номер мобільного телефону для кафедри і структурного підрозділу, має належати українському телефонному коду “+380”. Відповідно було створено тригер, який проводить перевірку, чи починається номер мобільного телефону з цього коду. Програмний код тригерної функції забезпечений перевіркою коректності даних. Якщо номер телефону буде інакшим, то тригерною функцією передбачено виклик виключної ситуації та оповіщення користувача про неможливість вставки даних. Код тригерної функції наведено у лістингу 3.3.

```

CREATE OR REPLACE FUNCTION validationPhoneNumberTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.phoneNumber LIKE '+380%' THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Мобільний номер має належати до
українського оператора зв'язку і мати код +380.';
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

```

### Лістинг 3.3 — Тригерна функція перевірки мобільного номеру

Оскільки і кафедра, і структурний підрозділ мають власні мобільні номери телефону, то виникає необхідність прив'язки цієї тригерної функції як

для таблиці “Chair”, так і для таблиці “StructureDepartment”. Створимо тригер через CREATE TRIGGER та прив’яжемо функцію до обох таблиць. Відповідний текст програмного коду показано у лістингу 3.4.

```
CREATE TRIGGER strDepPhoneValidationTrigger
BEFORE INSERT ON StructureDepartment
FOR EACH ROW
EXECUTE FUNCTION validationPhoneNumberTriggerFunction();

CREATE TRIGGER chairPhoneValidationTrigger
BEFORE INSERT ON Chair
FOR EACH ROW
EXECUTE FUNCTION validationPhoneNumberTriggerFunction();
```

Лістинг 3.4 — Створення тригеру для таблиць “Chair” і “StructureDepartment”

Всі структурні підрозділи та кафедри мають власні електронні пошти. Як відомо студентам теж надається електронна пошта під доменом університету. Тому було розроблено тригер, який перевіряє вхідні дані на те, чи належить пошта домену “@kpi.ua”. Відповідна тригерна функція перевіряє атрибут таблиць для “StructureDepartment” і “Chair”. Програмний текст коду наведено у лістингу 3.5.

```
CREATE OR REPLACE FUNCTION checkEmailTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.email LIKE '%@kpi.ua' THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Пошта структурного підрозділу або
кафедри має належати до домену університету @kpi.ua';
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
```

```
$$ LANGUAGE plpgsql;
```

### Лістинг 3.5 — Тригерна функція валідації електронної пошти

Тепер необхідно створити тригер для цих таблиць із використанням інструментів системи керування базою даних. Відповідні запити для таблиць “StructureDepartment” і “Chair” наведено у лістингу 3.6.

```
CREATE TRIGGER strDepEmailValidationTrigger
BEFORE INSERT ON StructureDepartment
FOR EACH ROW
EXECUTE FUNCTION checkEmailTriggerFunction();
```

```
CREATE TRIGGER chairEmailValidationTrigger
BEFORE INSERT ON Chair
FOR EACH ROW
EXECUTE FUNCTION checkEmailTriggerFunction();
```

### Лістинг 3.6 — Створення тригерів для таблиць “StructureDepartment” і “Chair”

Кожний підрозділ і кафедра, яка в нього входить, мають власні веб-ресурси, що зареєстровані під веб-доменом. Оскільки ці структури мають офіційний характер, то і веб-домен також має належати до університету, якому вони належать. Зокрема це домен “edu.kpi.ua”, який перевірятиметься тригерною функцією, код якої наведений у лістингу 3.7.

```
CREATE OR REPLACE FUNCTION websiteValidationTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.website LIKE '%.edu.kpi.ua' THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Веб-сайт кафедри або структурного
підрозділу має функціонувати під доменом університету .edu.kpi.ua';
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
```

```
END;
$$ LANGUAGE plpgsql;
```

### Лістинг 3.7 — Тригерна функція перевірки домену веб-сайту

Оскільки і кафедра, і підрозділ мають веб-ресурси, то тригер потрібно створити для їх таблиць у базі даних. Зокрема, мова йде про таблиці “Chair” та “StructureDepartment”. Створення тригерів для цих таблиць наведено у лістингу 3.8.

```
CREATE TRIGGER chairWebsiteValidationTrigger
BEFORE INSERT ON Chair
FOR EACH ROW
EXECUTE FUNCTION websiteValidationTriggerFunction();

CREATE TRIGGER strDepWebsiteValidationTrigger
BEFORE INSERT ON StructureDepartment
FOR EACH ROW
EXECUTE FUNCTION websiteValidationTriggerFunction();
```

### Лістинг 3.8 — Створення тригерів для таблиць “Chair” та “StructureDepartment”

У лістингу 3.9 наведений код тригерної функції, що перевіряє дату вступу студента. Наразі поточним роком є 2023, то дата вступу студента з врахуванням його навчання на бакалавраті та магістратурі не має перевищувати період у шість років. Функція перевіряє чи різниця між поточною датою і датою вступу студента не перевищує вказану величину.

```
CREATE OR REPLACE FUNCTION studentEntryDateTriggerFunction()
RETURNS TRIGGER AS $$
DECLARE
    maxDifference INTEGER := 6;
BEGIN
    IF EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
NEW.dateOfEntry) < maxDifference THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Студент вже випустився з університету';
    END IF;
```

```

EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

```

### Лістинг 3.9 — Тригерна функція перевірки дати вступу

Тепер слід визначитися із таблицею, для якої ми створимо тригер з відповідною функцією. Оскільки мова йде про дату вступу, то логічно що мова йде про студента. Відповідною таблицею у нашій базі даних є “Student”. Тому тригер, який показано у лістингу 3.10, присвоюється саме для цієї таблиці.

```

CREATE TRIGGER studentEntryYearTrigger
BEFORE INSERT ON Student
FOR EACH ROW
EXECUTE FUNCTION studentEntryDateTriggerFunction();

```

### Лістинг 3.10 — Створення тригеру для таблиці “Student”

Кількість годин, що виділяється на один предмет також має бути обмежена відповідно до кредитів ЄКТС. Існують певні предмети, на які можуть бути виділені аж 6 кредитів. Як відомо один кредит вимірюється у 32 годинах. Тому потрібно перевірити чи кількість годин, яка виділяється на певний предмет входить у відрізок [0;192]. Відповідну перевірку виконує тригерна функція описана у лістингу 3.11. Якщо кількість годин задовільняє діапазон, то запис буде додано у таблицю, інакше утвориться виключна ситуація з повідомленням для користувача і запис у таблицю не буде додано.

```

CREATE OR REPLACE FUNCTION checkSubjectHoursTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.hours > 0 AND NEW.hours < 192 THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Кількість годин на предмет має входити
у діапазон від 0 до 192 академічних годин';
    END IF;

```



```

EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

```

### Лістинг 3.11 — Тригерна функція перевірки кількості годин

Також необхідно контролювати кількість годин для кожного з виду контролю. Наприклад, календарний контроль проводиться набагато довше ніж модульна контрольна робота. Тому виникає необхідність розробити відповідну тригерну функцію, яка перевірятиме кількість годин для кожного контролю. Відповідний програмний код тригерної функції наведено у лістингу 3.12.

```

CREATE OR REPLACE FUNCTION typeOfControlTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    NEW.hours := CASE
        WHEN NEW.typeOfControl = 'Laboratory work' THEN 2
        WHEN NEW.typeOfControl = 'Computer practicum' THEN 2
        WHEN NEW.typeOfControl = 'Module controlling work' THEN 2
        WHEN NEW.typeOfControl = 'Exam' THEN 4
        WHEN NEW.typeOfControl = 'Offset' THEN 4
        WHEN NEW.typeOfControl = 'Calendar control' THEN 10
        WHEN NEW.typeOfControl = 'Semester control' THEN 10
        WHEN NEW.typeOfControl = 'Practice' THEN 2
        ELSE 0
    END;

    IF NEW.hours = 0 THEN
        RAISE EXCEPTION 'Невідомий тип контролю';
    ELSE
        RETURN NEW;
    END IF;
END;

```

```

        END IF;
    EXCEPTION
        WHEN SQLSTATE 'P0001' THEN
            RETURN NULL;
    END;
    $$ LANGUAGE plpgsql;

```

Лістинг 3.12 — Тригерна функція перевірки кількості годин для кожного з виду контролів

Тепер слід прив'язати цю тригерну функцію до відповідної для нього таблиці “TypeOfControl” за допомогою засобів PostgreSQL. У лістингу 3.13 показано, як створювався тригер для цієї таблиці. Зокрема тригер створювався для перевірки перед вставкою(імпортом) даних.

```

CREATE TRIGGER typeOfControlTrigger
BEFORE INSERT ON typeOfControl
FOR EACH ROW
EXECUTE FUNCTION typeOfControlTriggerFunction();

```

Лістинг 3.13 — Створення тригера для таблиці “TypeOfControl”

#### 4. Представлення

У базі даних цієї курсової роботи було створено п'ять представлень. Представлення є досить корисним інструментом у базах даних, що зберігають в собі результати досить складних запитів. Вони є досить зручними і гнучкими, якщо один і той самий запит використовується декілька разів [8].

Першим представленням, яке було створено у базі, є список студентів, які належать академічній групі з назвою “НА-21”. Представлення було створено на основі запиту із поєднанням за допомогою ключового слова AS. У лістингу 4.1 наведено код створення цього представлення.

```

CREATE VIEW HA21Students AS
SELECT Student.id,
        Student.secondname || ' ' || Student.firstname || ' ' ||
Student.patronymic AS "Name"

```

```
FROM Student WHERE academicGroup = (SELECT id FROM AcademicGroup
WHERE name = 'HA-21');
```

#### Лістинг 4.1 — Створення представлення групи студентів

Наступне представлення відображатиме студентів певного структурного підрозділу, які склали екзамени, щонайменше на оцінку “В”. Для цього представлення було створено вибірку даних, де відображається повне ім’я студента, його академічна група, предмет який він складав та оцінка за екзамен. У якості структурного підрозділу було обрано Факультет інформатики та обчислювальної техніки. Код створення представлення наведено у лістингу 4.2.

```
CREATE VIEW FICSBestStudents AS
SELECT Student.secondname || ' ' || Student.firstname || ' ' ||
Student.patronymic AS "Student",
AcademicGroup.name AS "Group", (SELECT Subject.name FROM Subject
WHERE Subject.id = "Control".subject) AS "Subject", Mark.markECTS AS
"Mark"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN AcademicGroup ON AcademicGroup.id = Student.academicGroup
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN StructureDepartment ON StructureDepartment.id =
Chair.structureDepartment
JOIN Mark ON "Control".mark = Mark.id
WHERE StructureDepartment.id = (SELECT id FROM StructureDepartment
WHERE name = 'Faculty of Informatics and Computer Science')
AND "Control".typeofcontrol = (SELECT id FROM TypeOfControl WHERE
typeofcontrol = 'Exam')
AND Mark.score >= (SELECT MIN(score) FROM Mark WHERE markECTS =
'B');
```

#### Лістинг 4.2 — Створення представлення відмінників факультету

У лістингу 4.3 показано створення представлення, яке містить в своєму складі працівників окремого структурного підрозділу, які працюють у ньому більше 15 років. Для створення цього представлення проводилася вибірка

даних, де відбиралося повне ім'я працівника, його вчене звання, стаж роботи, де цей самий стаж чисельно більший за 15 і назва структурного підрозділу рівна визначені завчасно назві.

```
CREATE VIEW IPSAHighEmployee AS
SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' ||
Teacher.patronymic AS "Employee",
(SELECT educationalDegree FROM EducationalDegree WHERE id =
Teacher.educationalDegree) AS "Degree",
EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningworking) AS "Experience"
FROM Teacher
JOIN Chair ON Chair.id = Teacher.chair
JOIN StructureDepartment ON Chair.structuredepartment =
StructureDepartment.id
WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningworking) >= 15
AND StructureDepartment.id = (SELECT id FROM StructureDepartment
WHERE name = 'Institute for Applied Systems Analysis')
```

Лістинг 4.3 — Створення представлення працівників із стажем роботи більшим за 15 років

Наступне представлення зберігає в своєму складі всі типи контрольних заходів, які проводилися по певному предмету. Є окремий предмет, по якому проводяться контрольні заходи: практичні заняття, модульні контрольні роботи, екзамени або заліки. Також врахуємо, що на певних кафедрах з предмету може проводитися екзамен, а на інших — залік. У лістингу 4.4 наведено тексти програмного коду для створення представлення.

```
CREATE VIEW HOUControlType AS
SELECT DISTINCT TypeOfControl.typeofcontrol, (SELECT Subject.name
FROM Subject WHERE "Control".subject = Subject.id)
FROM "Control"
JOIN TypeOfControl ON "Control".typeofcontrol = TypeOfControl.id
```

```
WHERE "Control".subject = (SELECT id FROM Subject WHERE name =
'History of Ukraine');
```

#### Лістинг 4.4 — Створення представлення контрольних заходів по певному предмету

У останньому представлені містяться дані про академічних кураторів Кафедри права. Для створення представлення проводиться вибірка інформації про повне ім'я куратора, назву групи та кафедру, де кафедра заздалегідь є визначеною. Код до створення цього представлення наведено у лістингу 4.5.

```
CREATE VIEW LawDepartmentCurators AS
SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' ||
Teacher.patronymic AS "Curator",
AcademicGroup.name AS "Group", Chair.name AS "Chair"
FROM AcademicGroup
JOIN Teacher ON Teacher.id = AcademicGroup.academiccurator
JOIN Chair ON Chair.id = AcademicGroup.chair

WHERE Chair.id = (SELECT id FROM Chair WHERE name = 'Law
Department');
```

#### Лістинг 4.5 — Створення представлення з академічними кураторами певної кафедри

### 5. SQL-запити

У ході виконання цієї курсової роботи було створено двадцять один DML запит. Тексти програмного коду, бізнес-правила та словесні описи запитів наведено нижче.

У аналізі предметного середовища ми визначили, що одним із завдань бази даних є перевірка оцінок студентів, а зокрема для певних контрольних заходів. Запит №1 проводить певну селекцію інформації, яку можна описати як відображення назви структурного підрозділу, повного імені студента, його академічної групи, предмету з якого він складав екзамен та результуючої

оцінки за цей екзамен, де унікальний ідентифікатор рівний десяти, оцінка менша за 60 (тобто недостатньо) і тип контролю відповідає екзамену. Код відповідного запиту відображено у лістингу 5.1, а результати його виконання на рисунку 5.1.

```
SELECT StructureDepartment.name AS "Department",
Student.firstName || ' ' || Student.secondName AS "Student name",
AcademicGroup.name AS "Group",
Subject.name AS "Discipline",
Mark.marECTS AS "Resulting Mark"
FROM StructureDepartment
JOIN Chair ON Chair.structuredepartment = StructureDepartment.id
JOIN AcademicGroup ON AcademicGroup.chair = Chair.id
JOIN Student ON Student.academicgroup = AcademicGroup.id
JOIN "Control" ON "Control".student = Student.id
JOIN Mark ON "Control".mark = Mark.id
JOIN TypeOfControl ON "Control".typeofcontrol = TypeOfControl.id
JOIN Subject ON "Control".subject = Subject.id
WHERE Chair.structuredepartment = 10 AND Mark.score < 60 AND
"Control".typeofcontrol = 4
AND EXTRACT(YEAR FROM "Control".dateOfControl) = 2022;
```

Лістинг 5.1 — Запит №1

	Department character varying (100) 🔒	Student name text 🔒	Group character varying (10) 🔒	Discipline character varying (50) 🔒	Resulting Mark character varying (5) 🔒
1	Institute of Mechanical Engineering	Doroteya Wintour	DD-45	Mathematics	E
2	Institute of Mechanical Engineering	Gaylord Sebring	DG-96	Pharmacy	E
3	Institute of Mechanical Engineering	Goran Collingworth	VZ-36	Graphic Design	F
4	Institute of Mechanical Engineering	Hodge Loakes	DG-96	Medicine	E
5	Institute of Mechanical Engineering	Janenna Baird	XE-12	Information Technology	F
6	Institute of Mechanical Engineering	Kamillah Harner	WQ-33	Social Sciences	E
7	Institute of Mechanical Engineering	Wilone Dilgarno	HL-96	Communication Studies	E

Рисунок 5.1 — Результат запиту №1

У аналізі предметного середовища вказано, що у складі структурних підрозділів функціонують кафедри, на який існують академічні групи із студентами. Кожна академічна група має скінченну кількість студентів. Запит №2 відобразить інформацію про кожен академічну групу і кількість її студентів

у заданому структурному підрозділі, відсортованому за спаданням за кількістю, інформація по академічних групах виводиться для структурного підрозділу з ID, що рівний 25. Код скрипту(запиту) наведено у лістингу 5.2, а результат виконання запиту відображено на рисунку 5.2. На рисунку буде відображена таблиця із результатом виконання запиту із використанням обраної СУБД.

```
SELECT
StructureDepartment.name,
AcademicGroup.name AS "Group",
COUNT(Student.id) AS "Amount of students"
FROM Student
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN StructureDepartment ON StructureDepartment.id =
Chair.structuredepartment

WHERE StructureDepartment.id = 25
GROUP BY AcademicGroup.name, StructureDepartment.name
ORDER BY "Amount of students" DESC;
```

Лістинг 5.2 — Запит №2

	name character varying (100) 🔒	Group character varying (10) 🔒	Amount of students bigint 🔒
1	Faculty of Chemical Technology	CB-26	14
2	Faculty of Chemical Technology	OZ-41	13
3	Faculty of Chemical Technology	YA-15	9
4	Faculty of Chemical Technology	MG-10	9
5	Faculty of Chemical Technology	XS-78	9
6	Faculty of Chemical Technology	VH-58	9
7	Faculty of Chemical Technology	GK-21	8
8	Faculty of Chemical Technology	DI-91	4

Рисунок 5.2 — Результат запиту №2

Концептуальна модель нашої бази даних передбачає працівників, що мають певний робочий стаж, що визначається за їх датою влаштування у структурний підрозділ. Наступний запит буде пов'язаний із штатним складом

певної кафедри. Необхідно відобразити назву кафедри, ім'я викладача та стаж його роботи для кафедри з ID, що рівний 66 і стажем роботи працівника не менше за 10 років. Скрипт запиту відображений у лістингу 5.3, а результат його виконання на рисунку 5.3.

```
SELECT Chair.name, Teacher.firstName || ' ' || Teacher.lastName,
       EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningWorking) AS Experience
FROM Chair
JOIN Teacher ON Teacher.chair = Chair.id
WHERE Chair = 66 AND
       EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningWorking) >= 10;
```

Лістинг 5.3 — Запит №3

	name character varying (50)	?column? text	experience numeric
1	Network Engineering Department	Trever Duffil	20
2	Network Engineering Department	Celia Shew	27
3	Network Engineering Department	Cinderella Arrington	20
4	Network Engineering Department	Alic Briance	19
5	Network Engineering Department	Gayel Gorgen	29
6	Network Engineering Department	Kincaid Kauscher	32
7	Network Engineering Department	Gerry Ajsik	30
8	Network Engineering Department	Celestina Jurzyk	16
9	Network Engineering Department	Gabbie Cosslett	19
10	Network Engineering Department	Sunny Ruseworth	19

Рисунок 5.3 — Результат запиту №3

Кожний з вид контролів супроводжується викладачем, який оцінює студентів. Таку характеристику ми надали для контролю в описі предметного середовища нашої бази даних. Наступний запит описується відображенням назви предмету, повного імені викладача, дати контролю та його типу, де продемонструються предмети, що викладаються викладачем “X” протягом весняного календарного контролю 2023 року. У ході вибірки необхідної інформації встановимо межі дати для визначення саме весняного контролю і



2023 рік, а також через тип контролю вкажемо про необхідність відображення даних саме для календарного контролю. Код запити відображено у лістингу 5.4, а результат його виконання на рисунку 5.4.

```
SELECT DISTINCT Subject.name,
Teacher.firstName || ' ' || Teacher.lastName AS "Teacher name",
"Control".dateOfControl,
(SELECT typeofControl FROM TypeOfControl WHERE id = (SELECT id FROM
TypeOfControl WHERE typeofControl = 'Calendar control'))
FROM "Control"
JOIN Subject ON "Control".subject = Subject.id
JOIN Teacher ON "Control".teacher = Teacher.id
WHERE Teacher.id = 55 AND
"Control".typeofControl = (SELECT id FROM TypeOfControl WHERE
typeofControl = 'Calendar control')
AND EXTRACT(YEAR FROM "Control".dateOfControl) = 2023 AND
EXTRACT(MONTH FROM "Control".dateOfControl) = 3;
```

Лістинг 5.4 — Запит №4

	name character varying (50)	Teacher name text	dateofcontrol date	typeofcontrol character varying (50)
1	Humanities	Trumann Aldhouse	2023-03-18	Calendar control
2	International Relations	Trumann Aldhouse	2023-03-16	Calendar control
3	Neuroscience	Trumann Aldhouse	2023-03-16	Calendar control

Рисунок 5.4 — Результат запити №4

За кожен дисципліну студент отримує атестацію або неатестацію протягом календарного контролю. Цю атестацію отримує певна кількість студентів. Відповідні визначення для контролів ми навели у аналізі предметної області. Запит №5 має наступний словесний опис: “Відобразити унікальний ідентифікатор дисципліни, її назву та кількість студентів, що задовільно склали по ній осінній календарний контроль 2022 року. Результати запити показано на рисунку 5.5, а його програмний код у вигляді SQL запити показано у лістингу 5.5.

```
WITH CalendarControl2022 AS(
```

```

SELECT Subject.id AS idSubject,
       Subject.name AS nameSubject,
       COUNT(Student.id) AS "Amount of students"
FROM "Control"
JOIN Subject ON "Control".subject = Subject.id
JOIN Student ON "Control".student = Student.id
WHERE EXTRACT(YEAR FROM "Control".dateOfControl) = 2022 AND
EXTRACT(MONTH FROM "Control".dateOfControl) = 10
AND "Control".attestation = TRUE
GROUP BY Subject.id, "Control".typeOfControl,
"Control".dateOfControl
ORDER BY "Amount of students" DESC
)
SELECT CalendarControl2022.idSubject,
       CalendarControl2022.nameSubject, CalendarControl2022."Amount of
students"
FROM CalendarControl2022
WHERE CalendarControl2022."Amount of students" = (SELECT
MAX(CalendarControl2022."Amount of students") FROM CalendarControl2022);

```

Лістинг 5.5 — Запит №5

	idsubject bigint	namesubject character varying (50)	Amount of students bigint
1	67	Library Science	4

Рисунок 5.5 — Результат запиту №5

Кожний з викладачів в університеті має власне вчене звання, як ми описали в аналізі предметної області. Тому наступний запит має наступне словесне формулювання, а саме відобразити повне ім'я викладача, його вчене звання, назву структурного підрозділу та назву кафедри, де вчене звання викладача - це "Професор". Код до запиту показано у лістингу 5.6, а результат запити на рисунку 5.6.

```

SELECT Teacher.firstname || ' ' || Teacher.lastname || ' ' ||
Teacher.patronymic AS "Teacher name",

```

```

EducationalDegree.educationalDegree AS "Degree",
StructureDepartment.name AS "Structure Department",
Chair.name AS "Chair"
FROM StructureDepartment
JOIN Chair ON Chair.structureDepartment = StructureDepartment.id
JOIN Teacher ON Teacher.chair = Chair.id
JOIN EducationalDegree ON Teacher.educationalDegree =
EducationalDegree.id
WHERE EducationalDegree.id =
(SELECT EducationalDegree.id FROM EducationalDegree WHERE
EducationalDegree = 'Professor');

```

Лістинг 5.6 — Запит №6

	Teacher name text	Degree character varying (30)	Structure Department character varying (100)	Chair character varying (50)
1	Valerie Ellerey Gomersal	Professor	Projecting Institute Of Design	Education Department
2	Nelson Giacopello Langley	Professor	Polytechnic Lyceum	Environmental Planning Department
3	Broderic Frensche Snibson	Professor	Projecting Institute Of Artificial Intelligence	Media Studies Department
4	Guinna Di Carlo Nannoni	Professor	Institute of Special Communications and Information Protecti...	Sociology Department
5	Giselbert Sillett Reynoldson	Professor	Projecting Institute Of Computer Graphics	Sports Management Department
6	Conrado Howe Eariney	Professor	Faculty of Chemical Engineering	Astronomy Department
7	Gates Shildrake Braywood	Professor	Projecting Institute Of Design	Early Childhood Education Department
Total rows: 220 of 220		Query complete 00:00:00.108		Ln 86, Col 1

Рисунок 5.6 — Результат запиту №6

Як вже було визначено у описі предметного середовища, по кожному предмету проводиться певний тип контрольного заходу і виставляється певна оцінка. Відобразимо повне ім'я студента, його оцінку та назву предмету, де оцінка за модульну контрольну роботу з дисципліни “Історія України” входить в діапазон [25;40]. Програмний код до цього запити наведений у лістингу 5.7, а результат його виконання на рисунку 5.7.

```

SELECT Student.firstName || ' ' || Student.secondName || ' ' ||
Student.patronymic AS "Student",
Mark.score AS "Mark", Subject.name AS "Subject"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN Subject ON "Control".subject = Subject.id
JOIN Mark ON "Control".mark = Mark.id
WHERE

```

```

"Control".typeOfControl =
  (SELECT TypeOfControl.id FROM TypeOfControl WHERE
TypeOfControl.typeOfControl = 'Module controlling work')
  AND "Control".subject = (SELECT Subject.id FROM Subject WHERE
Subject.name = 'History of Ukraine')
  AND "Control".mark BETWEEN (SELECT Mark.id FROM Mark WHERE score =
25 AND isResultingMark = FALSE) AND
  (SELECT Mark.id FROM Mark WHERE score = 40 AND isResultingMark =
FALSE);

```

Лістинг 5.7 — Запит №7

	Student text	Mark numeric	Subject character varying (50)
1	Evangelia Triggs Arendt	29	History of Ukraine
2	Doug Potzold Valde	32	History of Ukraine
3	Lucita Eagleton Allawy	36	History of Ukraine
4	Nataline Tillerton Ilbert	37	History of Ukraine
5	Jeanna Hallihane Mol...	38	History of Ukraine

Рисунок 5.7 — Результат виконання запиту

Кожний студент складає різні типи контролів і в результаті отримує загальну оцінку за дисципліну. За всі його дисципліни можна порахувати середній бал і це буде його рейтинговим балом за семестр. Запит №8 відобразить найкращого студента в університеті з найбільшим середнім балом. Відбудеться вибірка даних, де відобразиться повне ім'я студента, група, середня оцінка та його структурний підрозділ, у вибірці потрібно обрати студента з найбільшим середнім балом за усі дисципліни. Результат запиту показано на рисунку 5.8, а код скрипту у лістингу 5.8

```

WITH UniversityAverageMark AS(
  SELECT Student.id AS "StudentID", Student.firstName || ' ' ||
Student.secondName || ' ' || Student.patronymic AS "Student",
  AcademicGroup.name AS "Group",
  AVG(score) AS "Average mark", StructureDepartment.name AS
"Structure Department"
  FROM "Control"

```

```

JOIN Student ON "Control".student = Student.id
JOIN Mark ON "Control".mark = Mark.id
JOIN AcademicGroup ON Student.academicgroup = AcademicGroup.id
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN StructureDepartment ON Chair.structuredepartment =
StructureDepartment.id
    GROUP BY Student.id, "Student", AcademicGroup.id,
StructureDepartment.name)
    SELECT UniversityAverageMark."StudentID",
UniversityAverageMark."Student", UniversityAverageMark."Group",
    MAX(UniversityAverageMark."Average mark") AS "Max mark",
UniversityAverageMark."Structure Department"
    FROM UniversityAverageMark WHERE UniversityAverageMark."Average
mark" =
    (SELECT MAX(UniversityAverageMark."Average mark") FROM
UniversityAverageMark)
    GROUP BY UniversityAverageMark."StudentID",
UniversityAverageMark."Student", UniversityAverageMark."Group",
    UniversityAverageMark."Structure Department";

```

Лістинг 5.8 — Запит №8

	StudentID bigint	Student text	Group character varying (10)	Max mark numeric	Structure Department character varying (100)
1	2280	Winifield Heinert Pammenter	QW-90	99.0000000000000000	Faculty of Applied Mathematics
2	2680	Arch Rubinovitsch Bevis	CE-36	99.0000000000000000	Institute of Energy Saving and Energy Managment

Рисунок 5.8 — Результат запиту №8

Тепер виконаємо запит, який нам покаже всіх академічних кураторів кафедри права, які працюють на ній більше ніж 5 років. У вибірці візьмемо повне ім'я викладача, групу та кафедру, а також досвід його роботи. Текст коду запити відображено у лістингу 5.9, а результат його виконання на рисунку 5.9.

```

SELECT AcademicGroup.name AS "Group",
Teacher.firstName || ' ' || Teacher.lastname AS "Curator",
Chair.name AS "Chair",
EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningworking) AS "Experience"

```

```

FROM AcademicGroup
JOIN Teacher ON AcademicGroup.academiccurator = Teacher.id
JOIN Chair ON Chair.id = AcademicGroup.chair
WHERE Chair.id = (SELECT Chair.id FROM Chair WHERE Chair.name =
'Law Department')
AND EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM
Teacher.beginningworking) >= 5;

```

Лістинг 5.9 — Запит №9

	Group character varying (10) 🔒	Curator text 🔒	Chair character varying (50) 🔒	Experience numeric 🔒
1	XU-84	Kalindi Keilloh	Law Department	20
2	FT-18	Bessy Gurry	Law Department	31

Рисунок 5.9 — Результат запиту №9

У випадку недостатньої оцінки під час написання екзамену студент має можливість закрити академічну заборгованість під час додаткової сесії. У запиті №10 покажемо таких студентів. Словесний опис цього запиту можна сформулювати так: “Показати студентів, предмет та оцінку з цього предмету в балах та у шкалі ЄКТС, якщо оцінка більше за 60, то відобразити, що студент склав на основній сесії, інакше він йде на додаткову сесію”. Код до цього запиту показано у лістингу 5.10, а результат на рисунку 5.10.

```

SELECT Student.firstname || ' ' || Student.secondname AS "Student",
Subject.name AS "Subject",
Mark.score AS "Mark",
Mark.markeCTS AS "ECTS",
CASE
    WHEN Mark.score >= 60 THEN 'Default session'
    ELSE 'Extra session'
END AS "Session"
FROM "Control"
JOIN Student ON Student.id = "Control".student
JOIN AcademicGroup ON Student.academicgroup = AcademicGroup.id
JOIN Mark ON "Control".mark = Mark.id
JOIN Subject ON "Control".subject = Subject.id

```



```

WHERE AcademicGroup.id = 125
AND "Control".typeOfControl = (SELECT TypeOfControl.id FROM
TypeOfControl WHERE TypeOfControl.typeOfControl = 'Exam')
GROUP BY "Student", Subject.name, Mark.marECTS, "Session",
Mark.score
ORDER BY Mark.score DESC;

```

Лістинг 5.10 — Запит №10

	Student text	Subject character varying (50)	Mark numeric	ECTS character varying (5)	Session text
1	Slade Stockman	Criminal Justice	100	A	Default session
2	Marion Rantoul	Public Administration	95	A	Default session
3	Muriel Silman	Environmental Science	82	B	Default session
4	Slade Stockman	Geology	78	B	Default session
5	Gerard Marion	Communication Studies	74	C	Default session
6	Renato Ilymanov	Chemistry	55	E	Extra session
7	Muriel Silman	Speech-Language Pathology	54	E	Extra session
Total rows: 9 of 9		Query complete 00:00:00.146			

Рисунок 5.10 — Результат запиту №10

Кожен студент має власну залікову книжку, де стоять його оцінки за екзамени та заліки. У відомості показано предмет і результуючу оцінку, яку студент заробив протягом контролю. Наступний запит опишемо наступним словесним формулюванням. Відобразити всі предмети із залікової книжки окремого студента, який навчається у структурному підрозділі “А”, на кафедрі “В”, у групі “С”. Текст скрипту наведений у лістингу 5.11, а результат його вииконання на рисунку 5.11.

```

SELECT Student.id AS "Student ID",
Student.secondname || ' ' || Student.firstname || ' ' ||
Student.patronymic AS "Student",
Subject.name AS "Subject",
Mark.score AS "Grade mark",
Mark.marECTS AS "Result mark",
Mark.nationalMark AS "National grade",
Chair.name AS "Chair",
StructureDepartment.name AS "Structure Department"

```

```

FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON AcademicGroup.chair = Chair.id
JOIN StructureDepartment ON StructureDepartment.id =
Chair.structuredepartment
JOIN Mark ON Mark.id = "Control".mark
JOIN Subject ON Subject.id = "Control".subject
WHERE StructureDepartment.id = 1 AND
"Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE
typeOfControl = 'Exam')
AND Student.id = 2816
AND Chair.id = 1;

```

Лістинг 5.11 — Запит №11

	Student ID bigint	Student text	Subject character varying (50)	Grade mark numeric	Result mark character varying (5)	National grade character varying (15)	Chair character varying (50)	Structure Department character varying (100)
1	2816	Karus Thia Lyddyard	Botany	46	E	Not enough	Computer Science Department	Institute of Aerospace Technologies
2	2816	Karus Thia Lyddyard	History	80	B	Good	Computer Science Department	Institute of Aerospace Technologies

Рисунок 5.11 — Результат запиту №11

У наступному запиті відобразимо всі можливі комбінації кафедр, на яких студент міг би вчитися. Використаємо для цього вибірку імені студента, назву кафедри та назву структурного підрозділу та декартовий добуток множин із використанням CROSS JOIN. Код до цього запиту показано у лістингу 5.12, а результат його виконання на рисунку 5.12.

```

SELECT Student.id, Student.secondname, StructureDepartment.name,
Chair.name FROM Student
CROSS JOIN StructureDepartment
JOIN Chair ON Chair.structuredepartment = StructureDepartment.id
WHERE Student.id = 1;

```

Лістинг 5.12 — Запит №12



	id bigint	secondname character varying (50)	name character varying (100)	name character varying (50)
1	1	Henstone	Institute of Aerospace Technologies	Computer Science Department
2	1	Henstone	Institute of Aerospace Technologies	Information Technology Department
3	1	Henstone	Institute of Aerospace Technologies	Electronics Department
4	1	Henstone	Institute of Nuclear and Thermal Power Engineering	Mathematics Department
5	1	Henstone	Institute of Nuclear and Thermal Power Engineering	Physics Department
6	1	Henstone	Institute of Nuclear and Thermal Power Engineering	Chemistry Department
7	1	Henstone	Institute of Energy Saving and Energy Managment	Biology Department
8	1	Henstone	Institute of Energy Saving and Energy Managment	History Department
9	1	Henstone	Institute of Energy Saving and Energy Managment	Geography Department
10	1	Henstone	Institute of Materials Science and Welding	Literature Department
11	1	Henstone	Institute of Materials Science and Welding	Art Department
12	1	Henstone	Institute of Materials Science and Welding	Music Department
13	1	Henstone	Institute of Postgraduate Education	Engineering Department
14	1	Henstone	Institute of Postgraduate Education	Architecture Department
15	1	Henstone	Institute of Postgraduate Education	Economics Department
16	1	Henstone	Institute for Applied Systems Analysis	Business Department
Total rows: 100 of 100    Query complete 00:00:00.111				

Рисунок 5.12 — Результат запиту №12

Наступний запит формується на основі атестацій студентів, що складали календарні контролі протягом декількох курсів. Запит №13 супроводжується формулюванням, що необхідно відобразити студентів, що отримали атестацію по календарному контролю як в 2022 році, так і в 2023 році у конкретному структурному підрозділі. Результати цього запиту показано на рисунку 5.13, а його текст коду в лістингу 5.13.

```

WITH AtestatedStudents AS(
  WITH ResultingAttestation AS(
    SELECT Student.id AS "Student ID",Student.firstname || ' ' ||
Student.secondname AS "Student",
      Subject.name AS "Subject", "Control".attestation AS
"Attestation",
        "Control".dateofcontrol AS "Date", TypeOfControl.typeofcontrol
AS "Control"
    FROM "Control"
    JOIN Student ON Student.id = "Control".student
    JOIN Subject ON Subject.id = "Control".subject
    JOIN TypeOfControl ON TypeOfControl.id =
"Control".typeofcontrol
    JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup

```

```

JOIN Chair ON AcademicGroup.chair = Chair.id
JOIN StructureDepartment ON StructureDepartment.id =
Chair.structuredepartment
WHERE StructureDepartment.id = 10 AND "Control".typeOfControl
= (SELECT id FROM TypeOfControl WHERE typeOfControl = 'Calendar control')
) SELECT ResultingAttestation."Student ID" FROM
ResultingAttestation
WHERE EXTRACT(MONTH FROM ResultingAttestation."Date") = 10 AND
EXTRACT(YEAR FROM ResultingAttestation."Date") = 2023
AND ResultingAttestation."Attestation" = TRUE
INTERSECT
SELECT ResultingAttestation."Student ID" FROM ResultingAttestation
WHERE EXTRACT(MONTH FROM ResultingAttestation."Date") = 10 AND
EXTRACT(YEAR FROM ResultingAttestation."Date") = 2022
AND ResultingAttestation."Attestation" = TRUE)
SELECT "Student ID",Student.firstname || ' ' || Student.secondname
AS "Student"
FROM AtestatedStudents
JOIN Student ON Student.id = AtestatedStudents."Student ID";

```

Лістинг 5.13 — Запит №13

	Student ID bigint	Student text
1	117	Dorothea Grzesiewicz

Рисунок 5.13 — Результат запиту №13

Кожний студент є особою і має рахуватися у переліку в своєму структурному підрозділі. Наступний запит буде сформовано на основі твердження, що необхідно перевірити, чи існує певний студент в університеті, якщо існує, то відобразити інформацію про його персональні дані, кафедру, структурний підрозділ. Програмний текст коду цього запиту відображено у лістингу 5.14, а результат його виконання на рисунку 5.14.

```

SELECT Student.firstname || ' ' || Student.secondname AS "Student",
AcademicGroup.name AS "Group", Chair.name AS "Chair",
StructureDepartment.name AS "Structure Department"

```

```

FROM Student
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON AcademicGroup.chair = Chair.id
JOIN StructureDepartment ON StructureDepartment.id =
Chair.structuredepartment
WHERE EXISTS(SELECT 1 FROM Student WHERE firstname = 'Henry' AND
secondname = 'Henstone' AND patronymic = 'Sobtka')
AND Student.id = (SELECT id FROM Student WHERE firstname = 'Henry'
AND secondname = 'Henstone' AND patronymic = 'Sobtka');

```

Лістинг 5.14 — Запит №14

	Student text	Group character varying (10)	Chair character varying (50)	Structure Department character varying (100)
1	Henry Henstone	ME-34	Social Work Department	Publishing and Printing Institute

Рисунок 5.14 — Результат запиту №14

Певний викладач кафедри може викладати не одну кількість предметів. Зазвичай це декілька дисциплін. Тому наступний запит буде пов'язаний із тим скільки викладачі певної кафедри викладають дисциплін. Його словесний опис можна описати як “Відобразити ім'я викладача, назву кафедри на якій він працює та кількість предметів, яку він викладає, для кафедри “А”. Код цього запиту показано у лістингу 5.15, а результат його виконання на рисунку 5.15.

```

SELECT Chair.name AS "Chair",
Teacher.firstName || ' ' || Teacher.lastName AS "Teacher",
COUNT("Control".subject) AS "Amount of subjects"
FROM "Control"
JOIN Teacher ON Teacher.id = "Control".teacher
JOIN Chair ON Chair.id = Teacher.chair
WHERE Chair.id = 55
GROUP BY Chair.name, "Teacher"
ORDER BY "Amount of subjects" ASC;

```

Лістинг 5.15 — Запит №15

	Chair character varying (50)	Teacher text	Amount of subjects bigint
1	Optics Department	Iris Ferebee	16
2	Optics Department	Bank Monksfield	19
3	Optics Department	Ardith Farquarson	21
4	Optics Department	Xever Riglar	24
5	Optics Department	Kenon Jammet	25
6	Optics Department	Rand Bleacher	28

Рисунок 5.15 — Результат виконання запиту №15

Наступний запит буде супроводжуватися відомостями про успішність студентів у результаті семестру. Покажемо студентів певного структурного підрозділу, у яких загальна оцінка з певного предмету менша за 90 балів. У вибірці даних візьмемо повне ім'я студента, назву предмету, оцінку, оцінку за ЄКТС та назву структурного підрозділу. Текст коду до цього запиту, а також результати його виконання показано у лістингу 5.16 та рисунку 5.16 відповідно.

```

SELECT Student.secondname AS "Student", Subject.name AS "Subject",
Mark.score AS "Mark",
Mark.markeCTS AS "ECTS", StructureDepartment.name AS "Department"
FROM "Control"
JOIN Student ON Student.id = "Control".student
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN Mark ON Mark.id = "Control".mark
JOIN StructureDepartment ON StructureDepartment.id =
Chair.structuredepartment
JOIN Subject ON "Control".subject = Subject.id
WHERE StructureDepartment.id = 10 AND
"Control".typeofControl = (SELECT id FROM TypeOfControl WHERE
typeofcontrol = 'Exam')
AND Subject.id = 7;

```

Лістинг 5.16 — Запит №16

	Student character varying (50)	Subject character varying (50)	Mark numeric	ECTS character varying (5)	Department character varying (100)
1	Darree	History	74	C	Institute of Mechanical Engineering
2	Thunders	History	68	C	Institute of Mechanical Engineering

Рисунок 5.16 — Результат запиту №16

У наступному запиті покажемо кафедру, тип структурного підрозділу до якого вона входить, а також загальну кількість викладачів, що на ній працюють. В цьому запиті використаємо COUNT для підрахунку кількості. Код до запиту показаний у лістингу 5.17, а результат його виконання на рисунку 5.17.

```
SELECT
    c.name AS "Chair",
    td.type "Type of department",
    COUNT(t.id) AS "Amount of teachers"
FROM Chair c
JOIN StructureDepartment sd ON c.structureDepartment = sd.id
JOIN TypeOfDepartment td ON sd.type = td.id
LEFT JOIN Teacher t ON c.id = t.chair
GROUP BY c.name, td.type;
```

Лістинг 5.17 — Запит №17

	Chair character varying (50)	Type of department character varying (50)	Amount of teachers bigint
1	Electronics Department	Research and Educational Institute	10
2	Health Sciences Department	Projecting Institute	13
3	Information Science Department	Projecting Institute	14
4	Linguistics Department	Research and Educational Institute	12
5	Physical Therapy Department	Research and Educational Institute	13
6	Agricultural Science Department	Faculty	7
7	Optics Department	Faculty	6
8	Information Technology Department	Research and Educational Institute	11
9	Psychology Department	Research and Educational Institute	6
10	Physical Education Department	Projecting Institute	7
11	Computer Engineering Department	Faculty	16
12	Animal Science Department	Faculty	14
13	Computer Science Department	Research and Educational Institute	10
14	Tourism Management Department	Faculty	11
15	Chemical Engineering Department	Faculty	10

Рисунок 5.17 — Результат виконання запиту №17



У наступному запиті відобразимо всі контактні дані, що мають кафедри кожного структурного підрозділу університету. Контактні дані складаються з номеру мобільного телефону, електронної пошти та веб-сайту. Тому запит сформуємо за словесним описом: “Для кафедр структурного підрозділу відобразити їх назву, мобільний телефон, електронну пошту та адресу веб-ресурсу”. Текст коду запиту показано у лістингу 5.18, а результат його виконання на рисунку 5.18.

```
SELECT StructureDepartment.name AS "Structure Department",
Chair.name AS "Chair", Chair.phoneNumber AS "Phone",
Chair.website AS "Website"
FROM StructureDepartment
JOIN Chair ON Chair.structureDepartment = StructureDepartment.id
WHERE StructureDepartment.id = 18;
```

Лістинг 5.18 — Запит №18

	Structure Department character varying (100)	Chair character varying (50)	Phone character varying (30)	Website character varying (50)
1	Faculty of Biotechnology and Biotechnology	Statistics Department	+380552345678	http://stats.edu.kpi.ua
2	Faculty of Biotechnology and Biotechnology	Actuarial Science Department	+380553456789	http://actuarial.edu.kpi.ua
3	Faculty of Biotechnology and Biotechnology	Econometrics Department	+380554567890	http://econometrics.edu.kpi.ua
4	Faculty of Biotechnology and Biotechnology	Tourism Management Department	+380599012345	http://tourism.edu.kpi.ua

Рисунок 5.18 — Результат виконання запиту №18

Певний викладач може мати будь-яке вчене звання, яке він може здобути в ході своєї наукової діяльності. У наступному запиті будуть показані всі можливі для викладача вчені звання, які він може отримати в університеті. Запит може бути сформований словесно наступним формулюванням: “Відобразити для певного викладача, всі можливі вчені звання, які він може отримати”. Код запиту показаний у лістингу 5.19, а результат його виконання на рисунку 5.19.

```
SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' ||
Teacher.patronymic AS "Teacher",
Teacher.chair AS "Chair", EducationalDegree.educationaldegree
FROM Teacher
CROSS JOIN EducationalDegree
```

```
WHERE Teacher.chair = 15;
```

Лістинг 5.19 — Запит №19

	Teacher text	Chair bigint	educationaldegree character varying (30)
1	Parkeson Hercule Moline...	15	Assistant
2	O'Longain Cassandra Co...	15	Assistant
3	Codron Wolfy Astman	15	Assistant
4	Fawlo Mitchel Ballinger	15	Assistant
5	Eunson Serena Pethick	15	Assistant
6	Wychard Fenelia Hymas	15	Assistant
7	Sobey Pauli Hatfull	15	Assistant
8	Reding Kingsly Laidler	15	Assistant
9	Jankiewicz Kaile Haslin	15	Assistant
10	Cricket Rosalind Filchakov	15	Assistant
11	Diver Leupold Allonby	15	Assistant
12	Benda Astrid Boulton	15	Assistant
13	Klemmt Tammy Feaverye...	15	Assistant
14	Balcers Dulce Hyne	15	Assistant
15	Gorler Merridie Nusch	15	Assistant
16	Adnam Sidnee Dain	15	Assistant
17	Parkeson Hercule Moline...	15	Associate Professor
18	O'Longain Cassandra Co...	15	Associate Professor
Total rows: 80 of 80		Query complete 00:00:00.103	

Рисунок 5.19 — Результат виконання запиту №19

У описі предметного середовища ми описали, що кожний предмет за кількістю кредитів ЄКТС має відведену на нього кількість годин. Тому в наступному запиті буде показано предмет та сумарна кількість академічних годин, яка відведена на його контрольні заходи. Текст коду до цього запиту показана у лістингу 5.20, а результат його виконання на рисунку 5.21.

```
SELECT
s.name AS "Subject",
COUNT(ctrl.id) AS "Amount of hours"
FROM Subject s
JOIN "Control" ctrl ON s.id = ctrl.subject
WHERE s.hours > 100
GROUP BY s.name
ORDER BY "Amount of hours" DESC;
```

Лістинг 5.20 — Запит №20



	Subject character varying (50) 	Amount of hours bigint 
1	Industrial Engineering	240
2	Paleontology	226
3	Environmental Science	225
4	Mechanical Engineering	223
5	Botany	220
6	Architecture	218
7	Law	218
8	Criminology	217
9	Music	214
10	Education	214
11	Veterinary Science	213
12	Linguistics	213
13	Sociology	212
14	Advanced Mathematics	211
15	Philology	209

Рисунок 5.20 — Результат виконання запиту

У наступному запиті будуть показані викладача, який має успішних студентів. Якщо сформувавши словесний опис до запиту, то можна описати запит наступним твердженням: “Відобразити викладача, студенти якого мають оцінки з його предметів більші за 90 балів”. Код до запиту показано у лістингу 5.21, а результат його виконання на рисунку 5.21.

```
SELECT Teacher.lastname AS "Teacher", Student.secondname AS
"Student",
Mark.score AS "Mark", Mark.markeCTS AS "ECTS"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN Teacher ON "Control".teacher = Teacher.id
JOIN Mark ON "Control".mark = Mark.id
WHERE "Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE
typeofcontrol = 'Exam')
GROUP BY Teacher.lastname, Student.secondname, Mark.score,
Mark.markeCTS
HAVING (Mark.score) >= 90;
```



Лістинг 5.21 — Запит №21

	Teacher character varying (50)	Student character varying (50)	Mark numeric	ECTS character varying (5)
1	Killerby	Lindhe	94	B+
2	Faichney	Chanders	96	A
3	Dufaire	Brownlea	97	A
4	Flaws	Sannes	95	A
5	Riep	Shaefer	95	A
6	Arnow	Postlewhite	94	B+
7	Etienne	Timmis	100	A
8	Mechi	Geraldi	92	B+
9	Duff	Rulten	95	A
10	Cuffe	Veale	97	A
11	Bryde	Metzel	92	B+
12	Klees	Dincke	100	A
13	Hollidge	Schoolcroft	92	B+
14	Cozby	Alfuso	100	A

Рисунок 5.21 — Результат виконання запиту №21

## 6. Оптимізація запитів

У ході виконання курсової роботи було проведено оптимізацію запитів для пришвидшення часу їх виконання. Для пришвидшення роботи вибірок інформації було використано індекси, що спрощують пошук у таблицях.

У лістингу 6.1 наведено код створення індексів, які оптимізують роботу практично усіх запитів, що наведені у підрозділі №5 “SQL-запити”.

```
CREATE INDEX idxStudentPersonalData ON Student(firstname,
secondname, patronymic)
```

```
CREATE INDEX idxTeacherPersonalData ON Teacher(firstname, lastname,
patronymic);
```

```
CREATE INDEX idxStudentAcademicGroup ON Student(academicgroup);
```

```
CREATE INDEX idxAcademicGroup ON AcademicGroup(id);
```

```
CREATE INDEX idxChairDepartment ON Chair(structuredepartment);
```

```
CREATE INDEX idxStructureDepartment ON StructureDepartment(id);
```

```
CREATE INDEX idxTeacherChair ON Teacher(chair);
```

```

CREATE INDEX idxChairID ON Chair(id);
CREATE INDEX idxControlID ON "Control"(id);
CREATE INDEX idxControlTeacher ON "Control"(teacher);
CREATE INDEX idxControlSubject ON "Control"(subject);
CREATE INDEX idxControlTypeOfControl ON "Control"(typeofcontrol);
CREATE INDEX idxControlMark ON "Control"(mark);
CREATE INDEX idxControlStudent ON "Control"(student);
CREATE INDEX idxStudentID ON Student(id);
CREATE INDEX idxTeacherID ON Teacher(id);
CREATE INDEX idxTypeOfControl ON TypeOfControl(id);
CREATE INDEX idxMarkID ON Mark(id);
CREATE INDEX idxSubjectID ON Subject(id);

```

### Лістинг 6.1 — Створення індексів для оптимізації запитів

Тепер покажемо, як позначилося створення індексів для наших запитів до і після їх виконання. На рисунках 6.1 та 6.2 показано роботу запиту №1 до оптимізації та після неї відповідно.

The screenshot shows a SQL IDE with a query editor and a results pane. The query is as follows:

```

5 SELECT StructureDepartment.name AS "Department",
6 Student.firstName || ' ' || Student.secondName AS "Student name",
7 AcademicGroup.name AS "Group",
8 Subject.name AS "Discipline",
9 Mark.markECTS AS "Resulting Mark"
10 FROM StructureDepartment
11 JOIN Chair ON Chair.structuredepartment = StructureDepartment.id
12 JOIN AcademicGroup ON AcademicGroup.chair = Chair.id
13 JOIN Student ON Student.academicgroup = AcademicGroup.id
14 JOIN "Control" ON "Control".student = Student.id
15 JOIN Mark ON "Control".mark = Mark.id
16 JOIN TypeOfControl ON "Control".typeofcontrol = TypeOfControl.id
17 JOIN Subject ON "Control".subject = Subject.id
18 WHERE Chair.structuredepartment = 10 AND Mark.score < 60 AND "Control".typeofControl = 4
19 AND EXTRACT(YEAR FROM "Control".dateOfControl) = 2022;
20

```

The results pane shows the following columns: Department, Student name, Group, Discipline, and Resulting Mark. The data is as follows:

Department	Student name	Group	Discipline	Resulting Mark
Institute of Mechanical Engineering	Uladis Foltner	DC AC	DC AC	1

A status bar at the bottom indicates: "Successfully run. Total query runtime: 273 msec. 7 rows affected."

Рисунок 6.1 — Робота запиту №1 до оптимізації

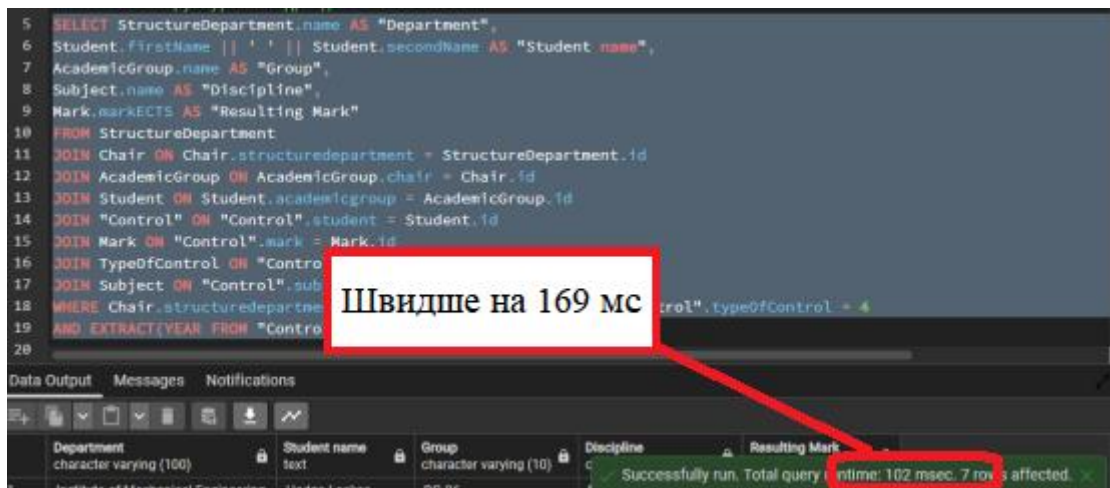


Рисунок 6.2 — Робота запиту №1 після оптимізації

Як бачимо робота індексів позитивно позначилася на виконанні запиту №1. Проте одного прикладу замало, тому наведемо ще декілька. Зокрема візьмемо другий запит та поглянемо на його роботу до оптимізації та після. Відповідні дані показано на рисунках 6.3 та 6.4 відповідно.

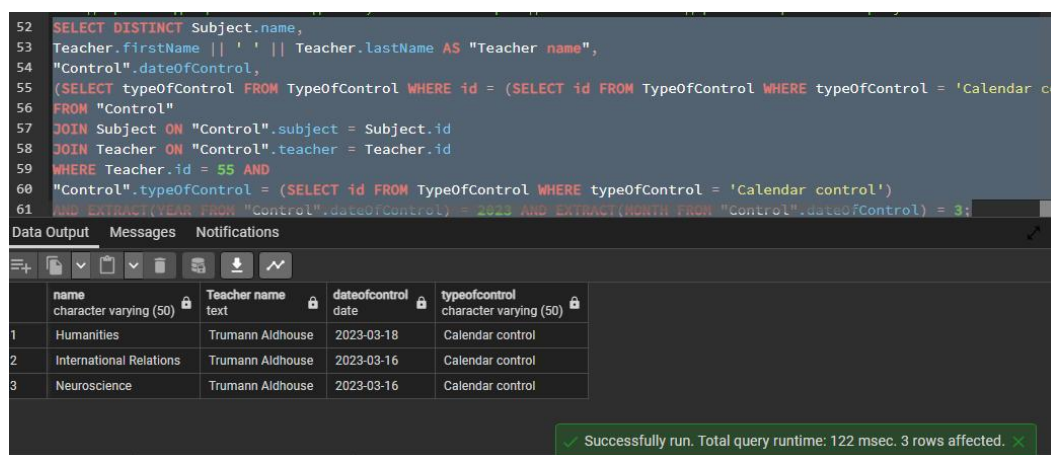


Рисунок 6.3 — Робота запиту №2 до оптимізації

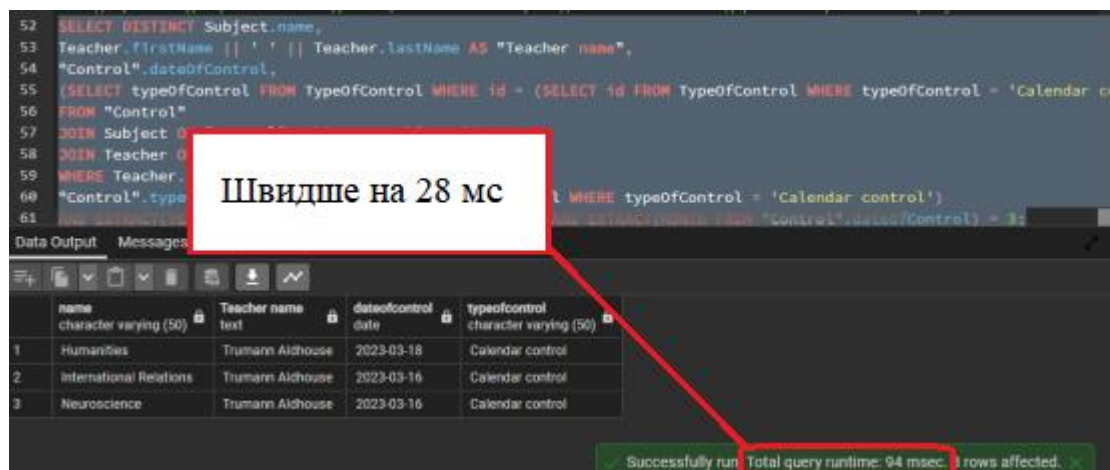


Рисунок 6.4 — Робота запиту №2 після оптимізації

Для того щоб переконатися у беззаперечній ефективності нашої оптимізації розглянемо ще один запит із нашої бази даних. На рисунках 6.5 та 6.6 показані приклади оптимізації для запиту №7, який описаний у підрозділі 5 “SQL-запити”.

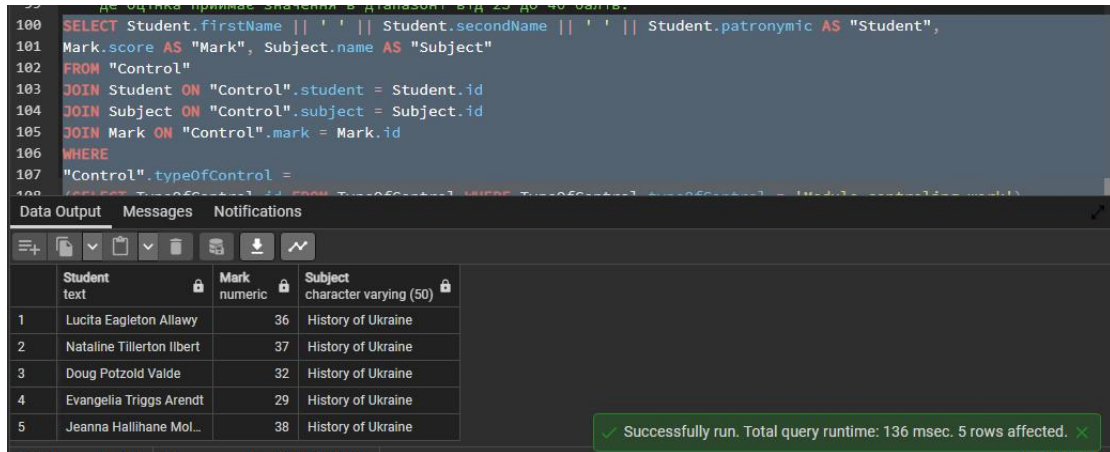


Рисунок 6.5 — Робота запиту №7 до оптимізації

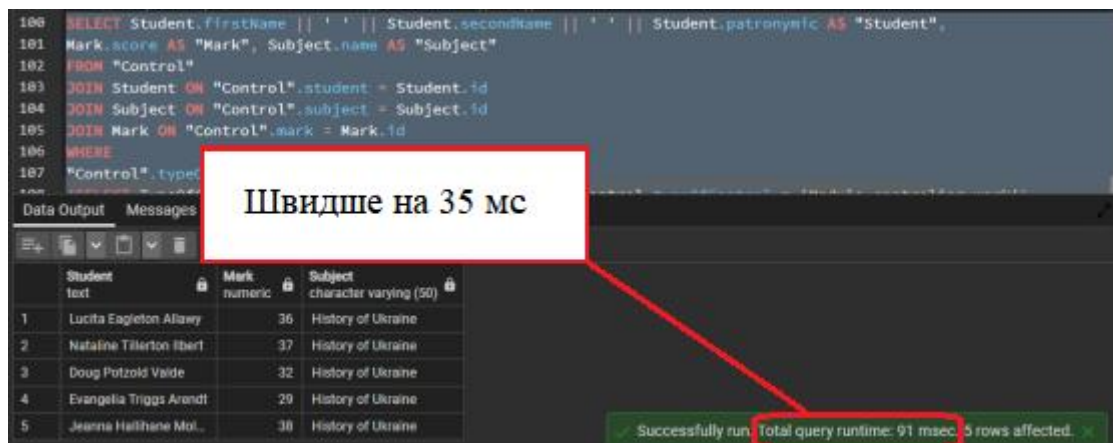


Рисунок 6.6 — Робота запиту №7 після оптимізації

## 8 ВИСНОВОК

У ході виконання курсової роботи з кредитного модуля “Бази даних” було проведено аналіз предметного середовища, розробку концептуальної моделі та імплементацію інфраструктури фізичної бази даних.

Передусім є важливим аналіз предметного середовища, який дозволяє виявити усі аспекти та чинники, які слід врахувати у подальшій розробці. Розробка концептуальної моделі є досить важливим етапом, оскільки вона фундаментує реалізацію фізичної моделі бази даних. Імплементація фізичної моделі бази даних вимагає від спеціаліста надзвичайно високих навичок та рівнів компетентностей для ефективного оперування даними у подальшому використанні.

У ході цієї курсової роботи було розроблено базу даних для регуляції екзаменаційної та залікової сесії, а також календарного та семестрового контролів. Виконання проєкту було досить корисним та ефективним для відточення навичок у використанні реляційних баз даних у майбутньому. Виконання проєкту супроводжувалося ґрунтовним аналізом предметної області обраної теми, формуванням бізнес-правил, розробкою концептуальної моделі та реалізації фізичної інфраструктури бази даних.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лузанов П. В., Рогов Є. В., Льовшин І. В. Postgres. Перше знайомство. Постгрес професійний, 2016, 180 с.
2. Автоматизована інформаційна система “Електронний кампус” КПІ ім. Ігоря Сікорського. URL: <https://campus.kpi.ua/> (дата звернення 30.11.2023).
3. Навчальна платформа Moodle. URL: <https://do.ipk.kpi.ua> (дата звернення 30.11.2023).
4. Навчальна платформа Human. URL: <https://www.human.ua/> (дата звернення 30.11.2023).
5. Монтицький Н. О. // GitHub Repository. URL: <https://github.com/NazaryiMontytskyi/DataBaseCourseWork> (дата звернення 25.12.2023).
6. PostgreSQL 16 // Документація зі створення тригерів. URL: <https://www.postgresql.org/docs/current/sql-createtrigger.html> (дата звернення 30.11.2023).
7. PostgreSQL 16 // Документація зі створення функцій та операторів. URL: <https://www.postgresql.org/docs/current/functions.html> (дата звернення 30.11.2023).
8. PostgreSQL 16 // Документація зі створення представлень. URL: <https://www.postgresql.org/docs/current/sql-createview.html> (дата звернення 30.11.2023).
9. A Guide to Building a Free Database for Education in 11 Simple Steps. URL: <https://www.knack.com/blog/database-for-education> (дата звернення 28.11.2023).

## ДОДАТОК А ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ТАБЛИЦЬ

--Створюємо тип структурного підрозділу, щоб надалі пов'язати його із сутністю

--структурного підрозділу

```
CREATE TABLE TypeOfDepartment(
    id BIGSERIAL PRIMARY KEY NOT NULL,
    type VARCHAR(50) NOT NULL
);
```

--Створюємо таблицю структурного підрозділу і пов'язуємо її з таблицею

--типу структурного підрозділу

```
CREATE TABLE StructureDepartment(
    id BIGSERIAL PRIMARY KEY NOT NULL,
    name VARCHAR(100) NOT NULL,
    phoneNumber VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL,
    website VARCHAR(50) NOT NULL,
    type BIGSERIAL NOT NULL REFERENCES TypeOfDepartment(id)
);
```

--Створемо таблицю кафедри, яку прив'яжемо до конкретного структурного підрозділу,

--в межах якого вона функціонує

```
CREATE TABLE Chair(
    id BIGSERIAL PRIMARY KEY NOT NULL,
    name VARCHAR(50) NOT NULL,
    phoneNumber VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL,
    website VARCHAR(50) NOT NULL,
    structureDepartment BIGSERIAL NOT NULL REFERENCES StructureDepartment(id)
);
```

--Створюємо таблицю наукових ступенів, щоб згодом прив'язати її до науково-педагогічного працівника

```
CREATE TABLE EducationalDegree(
    id BIGSERIAL PRIMARY KEY NOT NULL,
    educationalDegree VARCHAR(30) NOT NULL
);
```

--Створюємо таблицю вчителя до якої прив'язуємо кафедру на якій він викладає та  
 --науковий ступінь який він має

```
CREATE TABLE Teacher(
    id BIGSERIAL PRIMARY KEY NOT NULL,
    firstName VARCHAR(50) NOT NULL,
    lastName VARCHAR(50) NOT NULL,
    patronymic VARCHAR(50) NOT NULL,
    beginningWorking DATE NOT NULL,
    chair BIGSERIAL NOT NULL REFERENCES Chair(id),
    educationalDegree BIGSERIAL NOT NULL REFERENCES EducationalDegree(id)
);
```

--Створимо таблицю сутності академічної групи, яка буде посилатися на кафедру,  
 --в межах якої вона функціонує та на викладача, який є академічним куратором групи

```
CREATE TABLE AcademicGroup(
    id BIGSERIAL PRIMARY KEY NOT NULL,
    name VARCHAR(10) NOT NULL,
    chair BIGSERIAL NOT NULL REFERENCES Chair(id),
    academicCurator BIGSERIAL NOT NULL REFERENCES Teacher(id)
);
```

--Переходимо до початку імплементації типу контролю  
 --Спочатку створюємо сутність типу контролю, яка його визначатиме

```
CREATE TABLE TypeOfControl(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    typeOfControl VARCHAR(50) NOT NULL,
    hours INTEGER NOT NULL
);
```

--Створюємо тип оцінки для відображення у таблиці контролю

```
CREATE TABLE Mark(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    score NUMERIC NOT NULL,
    nationalMark VARCHAR(15) NOT NULL,
    markECTS VARCHAR(5) NOT NULL
);
```

--Створюємо таблицю типу предмету, щоб пов'язати її з типом предмету

```
CREATE TABLE TypeOfSubject(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    type VARCHAR(20) NOT NULL
);
```



--Створюємо таблицю предмету, у яку агрегуємо тип предмету

```
CREATE TABLE Subject(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    name VARCHAR(50) NOT NULL,
    hours INTEGER NOT NULL,
    typeOfSubject BIGSERIAL NOT NULL REFERENCES TypeOfSubject(id)
);
```

--Створюємо таблицю студента, у яку агрегуємо групу

```
CREATE TABLE Student(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    firstName VARCHAR(50) NOT NULL,
    secondName VARCHAR(50) NOT NULL,
    patronymic VARCHAR(50) NOT NULL,
    dateOfEntry DATE NOT NULL,
    academicGroup BIGSERIAL NOT NULL REFERENCES AcademicGroup(id)
);
```

--Створюємо тип контролю, у який агрегуємо

--студента, який проходить контроль

--оцінку, яку студент за контроль отримує

--тип контролю, який студент проходить

--викладача, який оцінює студента

--предмет, по якому контроль проводиться

```
CREATE TABLE "Control"(
    id BIGSERIAL NOT NULL PRIMARY KEY,
    attestation BOOL,
    dateOfControl DATE NOT NULL,
    typeOfControl BIGSERIAL NOT NULL REFERENCES TypeOfControl(id),
    mark INTEGER REFERENCES Mark(id),
    subject BIGSERIAL NOT NULL REFERENCES Subject(id),
    student BIGSERIAL NOT NULL REFERENCES Student(id),
    teacher BIGSERIAL NOT NULL REFERENCES Teacher(id)
);
```

## ДОДАТОК Б ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ОБМЕЖЕНЬ

--Вказуємо, що типи структурних підрозділів мають бути унікальними

```
ALTER TABLE TypeOfDepartment ADD CONSTRAINT uniqueDepartmentType UNIQUE(type);
```

--Вказуємо, що структурний підрозділ має мати унікальні:

--ім'я, номер мобільного, електронну пошту, та вебсайт

```
ALTER TABLE StructureDepartment ADD CONSTRAINT uniqueDepartmentName UNIQUE(name);
```

```
ALTER TABLE StructureDepartment ADD CONSTRAINT uniqueDepartmentPhoneNumber UNIQUE(phoneNumber);
```

```
ALTER TABLE StructureDepartment ADD CONSTRAINT uniqueDepartmentEmail UNIQUE(email);
```

```
ALTER TABLE StructureDepartment ADD CONSTRAINT uniqueDepartmentWebsite UNIQUE(website);
```

--Встановимо обмеження для кафедри, щоб вона мала унікальні:

--ім'я, номер мобільного, веб-сайт та електронну пошту

```
ALTER TABLE Chair ADD CONSTRAINT ChairUniqueName UNIQUE(name);
```

```
ALTER TABLE Chair ADD CONSTRAINT ChairUniquePhoneNumber UNIQUE(phoneNumber);
```

```
ALTER TABLE Chair ADD CONSTRAINT ChairUniqueWebsite UNIQUE(website);
```

```
ALTER TABLE Chair ADD CONSTRAINT ChairUniqueEmail UNIQUE(email);
```

--Встановимо обмеження для сутності AcademicGroup

--щоб вона мала унікальне ім'я

```
ALTER TABLE AcademicGroup ADD CONSTRAINT GroupUniqueName UNIQUE(name);
```

--Встановим обмеження для AcademicGroup, щоб кожна група

--мала унікального академічного куратора, тобто щоб утворився зв'язок

--один до одного

```
ALTER TABLE AcademicGroup ADD CONSTRAINT GroupUniqueCurator UNIQUE(academicCurator);
```

--Створимо умову, яка забезпечить унікальність всіх наукових ступенів

```
ALTER TABLE EducationalDegree ADD CONSTRAINT DegreeUniqueValue UNIQUE(educationalDegree);
```

--Створимо обмеження, де вкажемо, що тип предмету має бути унікальним

```
ALTER TABLE TypeOfSubject ADD CONSTRAINT SubjectUniqueType UNIQUE(type);
```

--Створимо обмеження, де вкажемо, що тип контролю має бути унікальним

```
ALTER TABLE TypeOfControl ADD CONSTRAINT ControlUniqueType UNIQUE(typeOfControl);
```

--Створимо обмеження, де вкажемо, що назва предмету має бути унікальною

```
ALTER TABLE Subject ADD CONSTRAINT SubjectUniqueName UNIQUE(name);
```

## ДОДАТОК В ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ТРИГЕРІВ

--Створюємо тригер, який перевіряє чи входить створена оцінка з контролю у діапазон 0-100

```
CREATE OR REPLACE FUNCTION validateMarkTriggerFunction() RETURNS TRIGGER AS $$
BEGIN
    IF NEW.score < 0 OR NEW.score > 100 THEN
        RAISE EXCEPTION 'Значення оцінки має входити у діапазон від 0 до 100 балів.';
    END IF;

    IF NEW.isResultingMark = TRUE THEN
        NEW.nationalMark := CASE
            WHEN NEW.score BETWEEN 0 AND 39 THEN 'Not allowed'
            WHEN NEW.score BETWEEN 40 AND 59 THEN 'Not enough'
            WHEN NEW.score BETWEEN 60 AND 64 THEN 'Enough'
            WHEN NEW.score BETWEEN 65 AND 74 THEN 'Satisfactory'
            WHEN NEW.score BETWEEN 75 AND 84 THEN 'Good'
            WHEN NEW.score BETWEEN 85 AND 94 THEN 'Very Good'
            ELSE 'Excellent'
        END;

        NEW.markeCTS := CASE
            WHEN NEW.score BETWEEN 0 AND 39 THEN 'F'
            WHEN NEW.score BETWEEN 40 AND 59 THEN 'E'
            WHEN NEW.score BETWEEN 60 AND 64 THEN 'D'
            WHEN NEW.score BETWEEN 65 AND 74 THEN 'C'
            WHEN NEW.score BETWEEN 75 AND 84 THEN 'B'
            WHEN NEW.score BETWEEN 85 AND 94 THEN 'B+'
            ELSE 'A'
        END;
    ELSE
        NEW.nationalMark := NULL;
        NEW.markeCTS := NULL;
    END IF;

    RETURN NEW;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;
```

--Створюємо тригер безпосередньо для таблиці Mark

```
CREATE TRIGGER markValidationTrigger
BEFORE INSERT ON Mark
FOR EACH ROW
EXECUTE FUNCTION validateMarkTriggerFunction();
```

```

--Створюємо тригер для перевірки коректності введення мобільного телефону
--відповідно до національного телефонного коду України +380
CREATE OR REPLACE FUNCTION validationPhoneNumberTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.phoneNumber LIKE '+380%' THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Мобільний номер має належати до українського оператора зв'язку і мати код +380.';
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

--Встановлюємо тригер на перевірку мобільного номера для структурного підрозділу та для кафедри
CREATE TRIGGER strDepPhoneValidationTrigger
BEFORE INSERT ON StructureDepartment
FOR EACH ROW
EXECUTE FUNCTION validationPhoneNumberTriggerFunction();

CREATE TRIGGER chairPhoneValidationTrigger
BEFORE INSERT ON Chair
FOR EACH ROW
EXECUTE FUNCTION validationPhoneNumberTriggerFunction();

--Створюємо тригерну функцію для перевірки коректності вводу електронної пошти, щоб
--вона відповідала шаблону '@kpi.ua'
CREATE OR REPLACE FUNCTION checkEmailTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
    IF NEW.email LIKE '%@kpi.ua' THEN
        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Пошта структурного підрозділу або кафедри має належати до домену університету
        @kpi.ua';
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

```

```
CREATE TRIGGER strDepEmailValidationTrigger
BEFORE INSERT ON StructureDepartment
FOR EACH ROW
EXECUTE FUNCTION checkEmailTriggerFunction();
```

```
CREATE TRIGGER chairEmailValidationTrigger
BEFORE INSERT ON Chair
FOR EACH ROW
EXECUTE FUNCTION checkEmailTriggerFunction();
```

--Створюємо тригерну функцію для перевірки домену веб-сайту структурного підрозділу  
--або кафедри. Домен має мати закінчення .edu.kpi.ua

```
CREATE OR REPLACE FUNCTION websiteValidationTriggerFunction()
RETURNS TRIGGER AS $$
BEGIN
```

```
    IF NEW.website LIKE '%.edu.kpi.ua' THEN
        RETURN NEW;
```

```
    ELSE
```

```
        RAISE EXCEPTION 'Веб-сайт кафедри або структурного підрозділу має функціонувати під доменом
університету .edu.kpi.ua';
```

```
    END IF;
```

```
EXCEPTION
```

```
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER chairWebsiteValidationTrigger
BEFORE INSERT ON Chair
FOR EACH ROW
EXECUTE FUNCTION websiteValidationTriggerFunction();
```

```
CREATE TRIGGER strDepWebsiteValidationTrigger
BEFORE INSERT ON StructureDepartment
FOR EACH ROW
EXECUTE FUNCTION websiteValidationTriggerFunction();
```

--Створюємо тригер, який перевіряє чи дата вступу студента не утворює  
--6 років різниці із поточною датою (бакалаврат + магістратура)

```
CREATE OR REPLACE FUNCTION studentEntryDateTriggerFunction()
RETURNS TRIGGER AS $$
```

```
DECLARE
```

```
    maxDifference INTEGER := 6;
```

```
BEGIN
```

```
    IF EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM NEW.dateOfEntry) < maxDifference
THEN
```

```

        RETURN NEW;
    ELSE
        RAISE EXCEPTION 'Студент вже випустився з університету';
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

```

```

CREATE TRIGGER studentEntryYearTrigger
BEFORE INSERT ON Student
FOR EACH ROW
EXECUTE FUNCTION studentEntryDateTriggerFunction();

```

--За законодавством України кожна дисципліна є кредитним модулем, яка має кількість годин

--відповідно до кредитів ЄКТС, яку на дисципліну було витрачено

```
CREATE OR REPLACE FUNCTION checkSubjectHoursTriggerFunction()
```

```
RETURNS TRIGGER AS $$
```

```
BEGIN
```

```
    IF NEW.hours > 0 AND NEW.hours < 192 THEN
```

```
        RETURN NEW;
```

```
    ELSE
```

```
        RAISE EXCEPTION 'Кількість годин на предмет має входити у діапазон від 0 до 192 академічних годин';
```

```
    END IF;
```

```
EXCEPTION
```

```
    WHEN SQLSTATE 'P0001' THEN
```

```
        RETURN NULL;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
CREATE TRIGGER checkSubjectHoursTrigger
```

```
BEFORE INSERT ON Subject
```

```
FOR EACH ROW
```

```
EXECUTE FUNCTION checkSubjectHoursTriggerFunction();
```

--Встановлюємо тригер, який перевіряє обмеження по кількості годин для типу контролю

```
CREATE OR REPLACE FUNCTION typeOfControlTriggerFunction()
```

```
RETURNS TRIGGER AS $$
```

```
BEGIN
```

```
    NEW.hours := CASE
```

```
        WHEN NEW.typeOfControl = 'Laboratory work' THEN 2
```

```
        WHEN NEW.typeOfControl = 'Computer practicum' THEN 2
```

```
        WHEN NEW.typeOfControl = 'Module controlling work' THEN 2
```

```
        WHEN NEW.typeOfControl = 'Exam' THEN 4
```

```

        WHEN NEW.typeOfControl = 'Offset' THEN 4
        WHEN NEW.typeOfControl = 'Calendar control' THEN 10
        WHEN NEW.typeOfControl = 'Semester control' THEN 10
        WHEN NEW.typeOfControl = 'Practice' THEN 2
        ELSE 0
    END;

    IF NEW.hours = 0 THEN
        RAISE EXCEPTION 'Невідомий тип контролю';
    ELSE
        RETURN NEW;
    END IF;
EXCEPTION
    WHEN SQLSTATE 'P0001' THEN
        RETURN NULL;
END;
$$ LANGUAGE plpgsql;

CREATE TRIGGER typeOfControlTrigger
BEFORE INSERT ON typeOfControl
FOR EACH ROW
EXECUTE FUNCTION typeOfControlTriggerFunction();

```

## ДОДАТОК Д   ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ФУНКЦІЙ ТА ЗБЕРЕЖЕНИХ ПРОЦЕДУР

---Процедура №1---

---Зміна кафедри викладача---

```
CREATE OR REPLACE PROCEDURE changeChairOfTeacher(teacherID INTEGER, newChairID INTEGER)
LANGUAGE plpgsql AS $$
DECLARE
    oldChair INTEGER;
BEGIN
    IF EXISTS(SELECT 1 FROM Teacher WHERE Teacher.id = teacherID) AND
    EXISTS(SELECT 1 FROM Chair WHERE Chair.id = newChairID) THEN
        SELECT chair INTO oldChair FROM Teacher WHERE Teacher.id = teacherID;
        UPDATE Teacher SET Chair = newChairID WHERE Teacher.id = teacherID;
        RAISE NOTICE 'Teacher with id % has change chair from % to %', teacherID, oldChair, newChairID;
    ELSE
        RAISE NOTICE 'Chair or teacher is incorrect';
    END IF;
END;
$$;
```

```
INSERT INTO Teacher(id,firstname, lastname, patronymic, chair, educationaldegree, beginningworking)
VALUES (1555,'Joseph', 'Biden', 'Robinett',5, 1, '12-10-2023');
CALL changeChairOfTeacher(1555, 6);
DELETE FROM Teacher WHERE Teacher.id = 1555;
```

---Функція №2---

---Відображення штатного розкладу структурного підрозділу

```
CREATE OR REPLACE FUNCTION queryOfStaffDepartment(department INTEGER)
RETURNS TABLE("Teacher" TEXT, "Chair" VARCHAR(150), "Department" VARCHAR(150)) AS $$
BEGIN
    RETURN QUERY SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' || Teacher.patronymic AS "Teacher",
        Chair.name AS "Chair", StructureDepartment.name AS "Department"
    FROM Teacher
    JOIN Chair ON Chair.id = Teacher.chair
    JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
    WHERE StructureDepartment.id = department;
END;
$$ LANGUAGE plpgsql;
```

```
DROP FUNCTION queryOfStaffDepartment(department INTEGER);
SELECT * FROM queryOfStaffDepartment(1);
```



---Процедура №3---

---Зміна персональних даних студента за його ID та новими даними---

```
CREATE OR REPLACE PROCEDURE updateStudentPersonalData(studentID INTEGER, newFirstName VARCHAR(50),
newSecondName VARCHAR(50), newPatronymic VARCHAR(50))
LANGUAGE plpgsql AS $$
BEGIN
    IF EXISTS(SELECT 1 FROM Student WHERE Student.id = studentID) THEN
        UPDATE Student SET firstname = newFirstName,
        secondname = newSecondName, patronymic = newPatronymic
        WHERE id = studentID;
        RAISE NOTICE 'Name of student with id % changed to % % %', studentID, newSecondName, newFirstName,
newPatronymic;
    ELSE
        RAISE NOTICE 'There is no student with such id';
    END IF;
END;
$$;
```

```
INSERT INTO Student(id, firstname, secondname, patronymic, academicgroup, dateofentry)
VALUES (5000, 'Donald', 'Trump', 'John', 34, '01-01-2021');
SELECT * FROM Student WHERE id = 5000;
CALL updateStudentPersonalData(5000, 'Michael', 'Pence', 'Richard');
SELECT * FROM Student WHERE id = 5000;
```

---Функція №4---

---Обрахувати кількість студентів для певного структурного підрозділу

---яка незадовільно склала семестровий контроль поточного року

```
CREATE OR REPLACE FUNCTION defineUnsuccessfulStudents(department INTEGER)
RETURNS INTEGER AS $$
DECLARE
    resultValue INTEGER;
BEGIN
    SELECT COUNT(Student.id) INTO resultValue
    FROM StructureDepartment
    JOIN Chair ON Chair.structureDepartment = StructureDepartment.id
    JOIN AcademicGroup ON AcademicGroup.chair = Chair.id
    JOIN Student ON Student.academicgroup = AcademicGroup.id
    JOIN "Control" ON Student.id = "Control".student
    WHERE StructureDepartment = department
    AND "Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE typeOfControl = 'Semester control')
    AND EXTRACT(YEAR FROM CURRENT_DATE) = EXTRACT(YEAR FROM "Control".dateOfControl)
    AND "Control".attestation = FALSE;

    RETURN resultValue;
END;
$$ LANGUAGE plpgsql;
```

```
SELECT * FROM defineUnsuccessfulStudents(5);
```

---Функція №5---

---Повернути таблицю, де будуть показані викладачі з вказаним

вченим званням та стажем роботи для конкретного структурного підрозділу

```
CREATE OR REPLACE FUNCTION queryDefinedStaffDepartment(department INTEGER, experineceValue INTEGER, degreeID
INTEGER)
```

```
RETURNS TABLE("Department" VARCHAR(150), "Teacher" TEXT, "Degree" VARCHAR(100), "Experience" NUMERIC) AS
$$
```

```
BEGIN
```

```
    RETURN QUERY SELECT StructureDepartment.name AS "Department",
```

```
    Teacher.lastname || ' ' || Teacher.firstname || ' ' || Teacher.patronymic AS "Teacher",
```

```
    EducationalDegree.educationaldegree AS "Degree",
```

```
    EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningworking) AS "Experience"
```

```
FROM StructureDepartment
```

```
JOIN Chair ON Chair.structuredepartment = StructureDepartment.id
```

```
JOIN Teacher ON Teacher.chair = Chair.id
```

```
JOIN EducationalDegree ON EducationalDegree.id = Teacher.educationalDegree
```

```
WHERE StructureDepartment.id = department
```

```
    AND EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningworking) >=
experineceValue
```

```
    AND Teacher.educationalDegree = degreeID;
```

```
END;
```

```
$$ LANGUAGE plpgsql;
```

```
SELECT * FROM queryDefinedStaffDepartment(
```

```
    (SELECT StructureDepartment.id FROM StructureDepartment WHERE StructureDepartment.name = 'Institute of
Aerospace Technologies')::INTEGER,
```

```
    12,
```

```
    (SELECT EducationalDegree.id FROM EducationalDegree WHERE educationaldegree = 'Senior Lecturer')::INTEGER
```

```
);
```

---Процедура №6---

---Виставлення оцінки студенту по певному виду контролю

---Вхідними параметрами будуть id студента, дата контролю

---id предмета, id викладача та оцінка, id контролю

```
CREATE OR REPLACE PROCEDURE createMarkForStudent(
```

```
    studentID INTEGER,
```

```
    requiredDate DATE,
```

```
    subjectID INTEGER,
```

```
    teacherID INTEGER,
```

```
    markScore INTEGER,
```

```
    controlID INTEGER
```

```

)
LANGUAGE plpgsql AS $$
DECLARE
    markID INTEGER;
BEGIN
    IF controlID != (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Semester control')
    AND controlID != (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Calendar control')
    THEN
        IF NOT EXISTS(SELECT 1 FROM Student WHERE Student.id = studentID) THEN
            RAISE NOTICE 'There is no such student in table.';
            RETURN;
        ELSIF NOT EXISTS(SELECT 1 FROM Subject WHERE Subject.id = subjectID) THEN
            RAISE NOTICE 'There is no such subject in table.';
            RETURN;
        ELSIF NOT EXISTS(SELECT 1 FROM Teacher WHERE Teacher.id = teacherID) THEN
            RAISE NOTICE 'There is no such teacher in table.';
            RETURN;
        ELSIF NOT EXISTS(SELECT 1 FROM TypeOfControl WHERE TypeOfControl.id = controlID) THEN
            RAISE NOTICE 'There is no such type of control in table.';
            RETURN;
        ELSIF markScore < 0 OR markScore > 100 THEN
            RAISE NOTICE 'Mark is not in required diapason.';
        ELSE
            SELECT id INTO markID FROM Mark WHERE Mark.score = markScore AND
Mark.isResultingMark = FALSE;
            INSERT INTO "Control"(student, dateofcontrol, subject, teacher, mark, typeofcontrol) VALUES
(studentID, requiredDate, subjectID, teacherID, markID, controlID);
            RAISE NOTICE 'New data about student control is created.';
            RAISE NOTICE 'Control id: %',
(SELECT id FROM "Control" WHERE student = studentID AND teacher = teacherID
AND subject = subjectID AND mark = markID AND typeofcontrol = controlID AND
dateofcontrol = requiredDate);
            RAISE NOTICE 'Student: %',
(SELECT Student.secondname || ' ' || Student.firstname || ' ' || Student.patronymic FROM Student
WHERE Student.id = studentID);
            RAISE NOTICE 'Teacher: %',
(SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' || Teacher.patronymic FROM Teacher WHERE
Teacher.id = teacherID);
            RAISE NOTICE 'Subject: %', (SELECT Subject.name FROM Subject WHERE Subject.id = subjectID);
            RAISE NOTICE 'Event: %', (SELECT TypeOfControl.typeofcontrol FROM TypeOfControl WHERE
TypeOfControl.id = controlID);
            RAISE NOTICE 'Mark: %', markScore;
        END IF;
    ELSE
        RAISE NOTICE 'You can not set numeric marks for calendar and semester control';
    END IF;
END IF;

```

END;

\$\$;

```
CALL createMarkForStudent(
    1, '2023-01-01', 3, 3, 15, 2
);
```

---Процедура №7---

---Залікова книжка студента---

CREATE OR REPLACE PROCEDURE studentRecordBook(studentID INTEGER)

LANGUAGE plpgsql AS \$\$

DECLARE

```
    recordBookCursor CURSOR FOR
    SELECT Subject.name AS "Subject",
           "Control".dateofcontrol AS "Date",
           Mark.score AS "Total score",
           Mark.markECTS AS "ECTS"
    FROM "Control"
    JOIN Subject ON Subject.id = "Control".subject
    JOIN Mark ON "Control".mark = Mark.id
    WHERE "Control".student = studentID
    AND "Control".typeofcontrol = (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Exam');
```

recordBookRecord RECORD;

BEGIN

OPEN recordBookCursor;

```
RAISE NOTICE 'Record book of %',
(SELECT Student.secondname || ' ' || Student.firstname || ' ' || Student.patronymic
FROM Student WHERE Student.id = studentID);
LOOP
```

```
    FETCH recordBookCursor INTO recordBookRecord;
    EXIT WHEN NOT FOUND;
    RAISE NOTICE 'Subject: % | Date: % | Score: % | ECTS: % ',
    recordBookRecord."Subject", recordBookRecord."Date",
    recordBookRecord."Total score", recordBookRecord."ECTS";
```

END LOOP;

CLOSE recordBookCursor;

END;

\$\$;

CALL studentRecordBook(55);

---Процедура №8---

---Пошук студента за його персональними параметрами

---Якщо відповідне ім'я існує, то відобразиться необхідна інформація про студента

---Якщо ні, то процедура відобразить відповідне повідомлення

```
CREATE OR REPLACE PROCEDURE findStudent(f_secondName VARCHAR(50), f_firstName VARCHAR(50), f_patronymic
VARCHAR(50))
```

```
LANGUAGE plpgsql AS $$
```

```
DECLARE
```

```
    studentRecord RECORD;
```

```
BEGIN
```

```
    IF EXISTS(SELECT 1 FROM Student WHERE firstname = f_firstname
```

```
        AND secondname = f_secondname
```

```
        AND patronymic = f_patronymic) THEN
```

```
        SELECT Student.secondName || ' ' || Student.firstName || ' ' || Student.patronymic AS "Student",
```

```
            AcademicGroup.name AS "Group", Chair.name AS "Chair", StructureDepartment.name AS
```

```
"Department"
```

```
            INTO studentRecord
```

```
        FROM Student
```

```
        JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
```

```
        JOIN Chair ON Chair.id = AcademicGroup.chair
```

```
        JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
```

```
        WHERE firstname = f_firstname
```

```
        AND secondname = f_secondname
```

```
        AND patronymic = f_patronymic;
```

```
        RAISE NOTICE 'Student: %', studentRecord."Student";
```

```
        RAISE NOTICE 'Group: %', studentRecord."Group";
```

```
        RAISE NOTICE 'Chair: %', studentRecord."Chair";
```

```
        RAISE NOTICE 'Structure Department: %', studentRecord."Department";
```

```
    ELSE
```

```
        RAISE NOTICE 'There is no such student in University.';
```

```
    END IF;
```

```
END;
```

```
$$;
```

```
DROP PROCEDURE findStudent(f_secondName VARCHAR(50), f_firstName VARCHAR(50), f_patronymic VARCHAR(50));
```

```
CALL findStudent('Henstone','Henry','Sobtko');
```

---Функція №9---

---Повертає кількість студентів, які незадовільно склали

---екзамени та підлягають відрахуванню

```
CREATE OR REPLACE FUNCTION countDismissalStudents()
```

```
RETURNS INTEGER AS $$
```

```
DECLARE
```

```
    resultVariable INTEGER;
```

```

BEGIN
    SELECT COUNT(DISTINCT Student.secondname || ' ' || Student.firstname || ' ' || Student.patronymic)
    INTO resultVariable
    FROM "Control"
    JOIN Student ON Student.id = "Control".student
    JOIN Mark ON Mark.id = "Control".mark
    WHERE "Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Exam')
    AND Mark.score < 60
    AND EXTRACT(YEAR FROM CURRENT_DATE) = EXTRACT(YEAR FROM "Control".dateofcontrol);
    RETURN resultVariable;
END;
$$ LANGUAGE plpgsql;

SELECT * FROM countDismissalStudents();

---Процедура №10---
---Відобразити всю необхідну контактну інформацію про університет
CREATE OR REPLACE PROCEDURE universityInfo()
LANGUAGE plpgsql AS $$
DECLARE
    amountOfFaculties INTEGER;
    amountOfInsistutes INTEGER;
    amountOfColleges INTEGER;
    amountOfProjInst INTEGER;
    departmentCursor CURSOR FOR
    SELECT StructureDepartment.name AS "Department",
    StructureDepartment.phoneNumber AS "Phone Number",
    StructureDepartment.website AS "Website",
    StructureDepartment.email AS "Email",
    TypeOfDepartment.type AS "Type"
    FROM StructureDepartment
    JOIN TypeOfDepartment ON TypeOfDepartment.id = StructureDepartment.type;
    departmentRecord RECORD;
BEGIN
    SELECT COUNT(*) INTO amountOfFaculties FROM StructureDepartment
    WHERE type = (SELECT id FROM TypeOfDepartment WHERE type = 'Faculty');
    SELECT COUNT(*) INTO amountOfInsistutes FROM StructureDepartment
    WHERE type = (SELECT id FROM TypeOfDepartment WHERE type = 'Research and Educational Institute');
    SELECT COUNT(*) INTO amountOfColleges FROM StructureDepartment
    WHERE type = (SELECT id FROM TypeOfDepartment WHERE type = 'Vocational College');
    SELECT COUNT(*) INTO amountOfProjInst FROM StructureDepartment
    WHERE type = (SELECT id FROM TypeOfDepartment WHERE type = 'Projecting Institute');

    RAISE NOTICE 'Information about University';
    RAISE NOTICE 'Faculties: %', amountOfFaculties;
    RAISE NOTICE 'Insitutes: %', amountOfInsistutes;

```

```
RAISE NOTICE 'Colleges: %', amountOfColleges;
RAISE NOTICE 'Projectin institues: %', amountOfProjInst;

OPEN departmentCursor;

LOOP
    FETCH departmentCursor INTO departmentRecord;
    EXIT WHEN NOT FOUND;
    RAISE NOTICE '-----';
    RAISE NOTICE '%', departmentRecord."Department";
    RAISE NOTICE '%', departmentRecord."Phone Number";
    RAISE NOTICE '%', departmentRecord."Website";
    RAISE NOTICE '%', departmentRecord."Email";
    RAISE NOTICE '%', departmentRecord."Type";
END LOOP;

CLOSE departmentCursor;

END;
$$;

CALL universityInfo();
```

## ДОДАТОК Е ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ПРЕДСТАВЛЕНЬ

---ПРЕДСТАВЛЕННЯ---

---Представлення №1---

---Представлення списку студентів, що числяться

---в групі HA-21

CREATE VIEW HA21Students AS

SELECT Student.id,

Student.secondname || ' ' || Student.firstname || ' ' || Student.patronymic AS "Name"

FROM Student WHERE academicGroup = (SELECT id FROM AcademicGroup WHERE name = 'HA-21');

SELECT \* FROM HA21Students;

---Представлення №2---

---Представлення студентів структурного підрозділу, що склали

---екзамени на оцінки щонайменше на В.

CREATE VIEW FICSBestStudents AS

SELECT Student.secondname || ' ' || Student.firstname || ' ' || Student.patronymic AS "Student",

AcademicGroup.name AS "Group", (SELECT Subject.name FROM Subject WHERE Subject.id = "Control".subject) AS "Subject",

Mark.markECTS AS "Mark"

FROM "Control"

JOIN Student ON "Control".student = Student.id

JOIN AcademicGroup ON AcademicGroup.id = Student.academicGroup

JOIN Chair ON Chair.id = AcademicGroup.chair

JOIN StructureDepartment ON StructureDepartment.id = Chair.structureDepartment

JOIN Mark ON "Control".mark = Mark.id

WHERE StructureDepartment.id = (SELECT id FROM StructureDepartment WHERE name = 'Faculty of Informatics and Computer Science')

AND "Control".typeofcontrol = (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Exam')

AND Mark.score >= (SELECT MIN(score) FROM Mark WHERE markECTS = 'B');

SELECT \* FROM FICSBestStudents;

---Представлення №3---

---Представлення викладачів структурного підрозділу, що мають

---робочий стаж більше 15 років

CREATE VIEW IPSAHighEmployee AS

SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' || Teacher.patronymic AS "Employee",

(SELECT educationalDegree FROM EducationalDegree WHERE id = Teacher.educationalDegree) AS "Degree",

EXTRACT(YEAR FROM CURRENT\_DATE) - EXTRACT(YEAR FROM Teacher.beginningworking) AS "Experience"

FROM Teacher

JOIN Chair ON Chair.id = Teacher.chair

JOIN StructureDepartment ON Chair.structuredepartment = StructureDepartment.id



```
WHERE EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningworking) >= 15
AND StructureDepartment.id = (SELECT id FROM StructureDepartment WHERE name = 'Institute for Applied Systems Analysis')
```

```
SELECT * FROM IPSAHighEmployee;
```

---Представлення №4---

---Відображення всіх типів контролів, які проводилися з певного предмету---

```
CREATE VIEW HOUControlType AS
```

```
SELECT DISTINCT TypeOfControl.typeofcontrol, (SELECT Subject.name FROM Subject WHERE "Control".subject = Subject.id)
FROM "Control"
```

```
JOIN TypeOfControl ON "Control".typeofcontrol = TypeOfControl.id
```

```
WHERE "Control".subject = (SELECT id FROM Subject WHERE name = 'History of Ukraine');
```

```
SELECT * FROM HOUControlType;
```

---Представлення №5---

---Відображення академічних кураторів груп окремої кафедри

```
CREATE VIEW LawDepartmentCurators AS
```

```
SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' || Teacher.patronymic AS "Curator",
```

```
AcademicGroup.name AS "Group", Chair.name AS "Chair"
```

```
FROM AcademicGroup
```

```
JOIN Teacher ON Teacher.id = AcademicGroup.academiccurator
```

```
JOIN Chair ON Chair.id = AcademicGroup.chair
```

```
WHERE Chair.id = (SELECT id FROM Chair WHERE name = 'Law Department');
```

```
SELECT * FROM LawDepartmentCurators;
```

## ДОДАТОК Ж ТЕКСТИ ПРОГРАМНОГО КОДУ ЗАПИТІВ

----###ЗАПИТ №1###----

--Визначити студентів структурного підрозділу X, які склали екзамени у 2022 році

--на незадовільну оцінку

--Візьмемо структурний підрозділ з id = 10

```
SELECT StructureDepartment.name AS "Department",
Student.firstName || ' ' || Student.lastName AS "Student name",
AcademicGroup.name AS "Group",
Subject.name AS "Discipline",
Mark.markECTS AS "Resulting Mark"
FROM StructureDepartment
JOIN Chair ON Chair.structuredepartment = StructureDepartment.id
JOIN AcademicGroup ON AcademicGroup.chair = Chair.id
JOIN Student ON Student.academicgroup = AcademicGroup.id
JOIN "Control" ON "Control".student = Student.id
JOIN Mark ON "Control".mark = Mark.id
JOIN TypeOfControl ON "Control".typeofcontrol = TypeOfControl.id
JOIN Subject ON "Control".subject = Subject.id
WHERE Chair.structuredepartment = 10 AND Mark.score < 60 AND "Control".typeofcontrol = 4
AND EXTRACT(YEAR FROM "Control".dateOfControl) = 2022;
```

----###ЗАПИТ №2###----

--Відобразити групи та кількість їх студентів структурного підрозділу

--X за спаданням (від більшого до меншого)

--Для прикладу візьмемо структурний підрозділ з id = 25

```
SELECT
StructureDepartment.name,
AcademicGroup.name AS "Group",
COUNT(Student.id) AS "Amount of students"
FROM Student
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
WHERE StructureDepartment.id = 25
GROUP BY AcademicGroup.name, StructureDepartment.name
ORDER BY "Amount of students" DESC;
```

----###ЗАПИТ №3###----

--Відобразити викладачів кафедри X, які викладають на цій

---кафедрі більше 10 років, їх імена, час роботи на кафедрі

```
SELECT Chair.name, Teacher.firstName || ' ' || Teacher.lastName,
EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningWorking) AS Experience
FROM Chair
JOIN Teacher ON Teacher.chair = Chair.id
```

```
WHERE Chair = 66 AND
EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningWorking) >= 10;
```

----###ЗАПИТ №4###----

--Відобразити дисципліни викладача X, по яким він проводив весняний календарний контроль 2023 року

```
SELECT DISTINCT Subject.name,
Teacher.firstName || ' ' || Teacher.lastName AS "Teacher name",
"Control".dateOfControl,
(SELECT typeOfControl FROM TypeOfControl WHERE id = (SELECT id FROM TypeOfControl WHERE typeOfControl =
'Calendar control'))
FROM "Control"
JOIN Subject ON "Control".subject = Subject.id
JOIN Teacher ON "Control".teacher = Teacher.id
WHERE Teacher.id = 55 AND
"Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE typeOfControl = 'Calendar control')
AND EXTRACT(YEAR FROM "Control".dateOfControl) = 2023 AND EXTRACT(MONTH FROM "Control".dateOfControl) = 3;
```

----###ЗАПИТ №5### (з підзапитом)---

---Відобразити дисципліну з найбільшою кількістю студентів, які склали по ній осінній

---календарний контроль 2022 року.

```
WITH CalendarControl2022 AS(
    SELECT Subject.id AS idSubject,
    Subject.name AS nameSubject,
    COUNT(Student.id) AS "Amount of students"
    FROM "Control"
    JOIN Subject ON "Control".subject = Subject.id
    JOIN Student ON "Control".student = Student.id
    WHERE EXTRACT(YEAR FROM "Control".dateOfControl) = 2022 AND EXTRACT(MONTH FROM
"Control".dateOfControl) = 10
    AND "Control".attestation = TRUE
    GROUP BY Subject.id, "Control".typeOfControl, "Control".dateOfControl
    ORDER BY "Amount of students" DESC
)
SELECT CalendarControl2022.idSubject, CalendarControl2022.nameSubject, CalendarControl2022."Amount of students"
FROM CalendarControl2022
WHERE CalendarControl2022."Amount of students" = (SELECT MAX(CalendarControl2022."Amount of students") FROM
CalendarControl2022);
```

----###ЗАПИТ №6###----

---Відобразити всіх професорів університету (ім'я, структурний підрозділ, кафедра та їх вчене звання)

```
SELECT Teacher.firstName || ' ' || Teacher.lastName || ' ' || Teacher.patronymic AS "Teacher name",
EducationalDegree.educationalDegree AS "Degree", StructureDepartment.name AS "Structure Department",
Chair.name AS "Chair"
```

```

FROM StructureDepartment
JOIN Chair ON Chair.structureDepartment = StructureDepartment.id
JOIN Teacher ON Teacher.chair = Chair.id
JOIN EducationalDegree ON Teacher.educationalDegree = EducationalDegree.id
WHERE EducationalDegree.id =
(SELECT EducationalDegree.id FROM EducationalDegree WHERE EducationalDegree = 'Professor');

```

----###ЗАПИТ №7###----

----Відобразити імена студентів, які склали модульну контрольну роботу з дисципліни "Історія України"

----де оцінка приймає значення в діапазоні від 25 до 40 балів.

```

SELECT Student.firstName || ' ' || Student.secondName || ' ' || Student.patronymic AS "Student",
Mark.score AS "Mark", Subject.name AS "Subject"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN Subject ON "Control".subject = Subject.id
JOIN Mark ON "Control".mark = Mark.id
WHERE
"Control".typeOfControl =
(SELECT TypeOfControl.id FROM TypeOfControl WHERE TypeOfControl.typeOfControl = 'Module controlling work')
AND "Control".subject = (SELECT Subject.id FROM Subject WHERE Subject.name = 'History of Ukraine')
AND "Control".mark BETWEEN (SELECT Mark.id FROM Mark WHERE score = 25 AND isResultingMark = FALSE) AND
(SELECT Mark.id FROM Mark WHERE score = 40 AND isResultingMark = FALSE);

```

----###ЗАПИТ №8###----

----Відобразити найкращого студента в університеті, що має найбільший середній бал

```

WITH UniversityAverageMark AS(
SELECT Student.id AS "StudentID", Student.firstName || ' ' || Student.secondName || ' ' || Student.patronymic AS "Student",
AcademicGroup.name AS "Group",
AVG(score) AS "Average mark", StructureDepartment.name AS "Structure Department"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN Mark ON "Control".mark = Mark.id
JOIN AcademicGroup ON Student.academicgroup = AcademicGroup.id
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN StructureDepartment ON Chair.structuredepartment = StructureDepartment.id
GROUP BY Student.id, "Student", AcademicGroup.id, StructureDepartment.name)
SELECT UniversityAverageMark."StudentID", UniversityAverageMark."Student", UniversityAverageMark."Group",
MAX(UniversityAverageMark."Average mark") AS "Max mark", UniversityAverageMark."Structure Department"
FROM UniversityAverageMark WHERE UniversityAverageMark."Average mark" =
(SELECT MAX(UniversityAverageMark."Average mark") FROM UniversityAverageMark)
GROUP BY UniversityAverageMark."StudentID", UniversityAverageMark."Student", UniversityAverageMark."Group",
UniversityAverageMark."Structure Department";

```

----###ЗАПИТ №9###----

----Відобразити всіх академічних кураторів груп кафедри Х, які працюють на цій кафедрі більше ніж 5 років

----Візьмемо Кафедру права ('Law Department')

```
SELECT AcademicGroup.name AS "Group",
Teacher.firstName || ' ' || Teacher.lastname AS "Curator",
Chair.name AS "Chair",
EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningworking) AS "Experience"
FROM AcademicGroup
JOIN Teacher ON AcademicGroup.academiccurator = Teacher.id
JOIN Chair ON Chair.id = AcademicGroup.chair
WHERE Chair.id = (SELECT Chair.id FROM Chair WHERE Chair.name = 'Law Department')
AND EXTRACT(YEAR FROM CURRENT_DATE) - EXTRACT(YEAR FROM Teacher.beginningworking) >= 5;
```

----###ЗАПИТ №10###----

----Відобразити сумарну оцінку студентів певної групи, яку потрібно відсортувати за

----спаданням по оцінкам. Якщо кількість балів студента не задовільна для виставлення

----оцінки по дисципліні, то відобразити у комірці "Додаткова сесія", а якщо задовільна,

----то відобразити "Основна сесія"

----В якості групи візьмемо групу з id 125

----В якості дисципліни візьмемо

```
SELECT Student.firstname || ' ' || Student.secondname AS "Student",
Subject.name AS "Subject",
Mark.score AS "Mark",
Mark.markECTS AS "ECTS",
CASE
    WHEN Mark.score >= 60 THEN 'Default session'
    ELSE 'Extra session'
END AS "Session"
FROM "Control"
JOIN Student ON Student.id = "Control".student
JOIN AcademicGroup ON Student.academicgroup = AcademicGroup.id
JOIN Mark ON "Control".mark = Mark.id
JOIN Subject ON "Control".subject = Subject.id
WHERE AcademicGroup.id = 125
AND "Control".typeOfControl = (SELECT TypeOfControl.id FROM TypeOfControl WHERE TypeOfControl.typeOfControl =
'Exam')
GROUP BY "Student", Subject.name, Mark.markECTS, "Session", Mark.score
ORDER BY Mark.score DESC;
```

----###ЗАПИТ №11###----

----Відобразити відомість про залікову книжку студента, який навчається

----у структурному підрозділі А, на кафедрі В, у групі С

```
SELECT Student.id AS "Student ID",
Student.secondname || ' ' || Student.firstname || ' ' || Student.patronymic AS "Student",
```

```

Subject.name AS "Subject",
Mark.score AS "Grade mark",
Mark.markECTS AS "Result mark",
Mark.nationalMark AS "National grade",
Chair.name AS "Chair",
StructureDepartment.name AS "Structure Department"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON AcademicGroup.chair = Chair.id
JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
JOIN Mark ON Mark.id = "Control".mark
JOIN Subject ON Subject.id = "Control".subject
WHERE StructureDepartment.id = 1 AND
"Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE typeOfControl = 'Exam')
AND Student.id = 2816
AND Chair.id = 1;

```

----###ЗАПИТ №12###----

----Відобразити всі можливі комбінації, де певний студент міг би обрати конкретний  
 ----структурний підрозділ та конкретну кафедру цього структурного підрозділу

```

SELECT Student.id, Student.secondname, StructureDepartment.name, Chair.name FROM Student
CROSS JOIN StructureDepartment
JOIN Chair ON Chair.structureDepartment = StructureDepartment.id
WHERE Student.id = 1;

```

----###ЗАПИТ №13###----

----Відобразити студентів структурного підрозділу, що були атестовані по  
 ----осінньому календарному контролю 2023 року і одночасно по календарному контролю 2022 року

```

WITH AtestedStudents AS(
WITH ResultingAttestation AS(
    SELECT Student.id AS "Student ID", Student.firstname || ' ' || Student.secondname AS "Student",
    Subject.name AS "Subject", "Control".attestation AS "Attestation",
    "Control".dateofcontrol AS "Date", TypeOfControl.typeofcontrol AS "Control"
    FROM "Control"
    JOIN Student ON Student.id = "Control".student
    JOIN Subject ON Subject.id = "Control".subject
    JOIN TypeOfControl ON TypeOfControl.id = "Control".typeofcontrol
    JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
    JOIN Chair ON AcademicGroup.chair = Chair.id
    JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
    WHERE StructureDepartment.id = 10 AND "Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE
typeOfControl = 'Calendar control')
) SELECT ResultingAttestation."Student ID" FROM ResultingAttestation

```

```

WHERE EXTRACT(MONTH FROM ResultingAttestation."Date") = 10 AND EXTRACT(YEAR FROM
ResultingAttestation."Date") = 2023
AND ResultingAttestation."Attestation" = TRUE
INTERSECT
SELECT ResultingAttestation."Student ID" FROM ResultingAttestation
WHERE EXTRACT(MONTH FROM ResultingAttestation."Date") = 10 AND EXTRACT(YEAR FROM
ResultingAttestation."Date") = 2022
AND ResultingAttestation."Attestation" = TRUE)
SELECT "Student ID", Student.firstname || ' ' || Student.secondname AS "Student"
FROM AtestatedStudents
JOIN Student ON Student.id = AtestatedStudents."Student ID";

```

----####Запит №14----

```

----Перевірити чи існує певний студент в університеті, якщо існує,
----то вивести інформацію про його ім'я, групу, кафедру та структурний підрозділ
SELECT Student.firstname || ' ' || Student.secondname AS "Student",
AcademicGroup.name AS "Group", Chair.name AS "Chair", StructureDepartment.name AS "Structure Department"
FROM Student
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON AcademicGroup.chair = Chair.id
JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
WHERE EXISTS(SELECT 1 FROM Student WHERE firstname = 'Henry' AND secondname = 'Henstone' AND patronymic =
'Sobtka')
AND Student.id = (SELECT id FROM Student WHERE firstname = 'Henry' AND secondname = 'Henstone' AND patronymic =
'Sobtka');

```

----####Запит №15----

```

----Для певної кафедри відобразити список викладачів та кількість предметів, яку вони викладають
----відсортувати цей список за зростанням
SELECT Chair.name AS "Chair",
Teacher.firstName || ' ' || Teacher.lastName AS "Teacher",
COUNT("Control".subject) AS "Amount of subjects"
FROM "Control"
JOIN Teacher ON Teacher.id = "Control".teacher
JOIN Chair ON Chair.id = Teacher.chair
WHERE Chair.id = 55
GROUP BY Chair.name, "Teacher"
ORDER BY "Amount of subjects" ASC;

```

----####Запит №16----

```

----Відобразити студентів структурного підрозділу X, підсумкові оцінки, по предмету У менші
----за 90 балів
SELECT Student.secondname AS "Student", Subject.name AS "Subject", Mark.score AS "Mark",
Mark.marECTS AS "ECTS", StructureDepartment.name AS "Department"

```

```

FROM "Control"
JOIN Student ON Student.id = "Control".student
JOIN AcademicGroup ON AcademicGroup.id = Student.academicgroup
JOIN Chair ON Chair.id = AcademicGroup.chair
JOIN Mark ON Mark.id = "Control".mark
JOIN StructureDepartment ON StructureDepartment.id = Chair.structuredepartment
JOIN Subject ON "Control".subject = Subject.id
WHERE StructureDepartment.id = 10 AND
"Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Exam')
AND Subject.id = 7;

```

----###Запит №17###----

----Відобразити кафедру, тип структурного підрозділу в якій вона перебуває  
 ----та кількість викладачів, які на цій кафедрі працюють

```

SELECT
  c.name AS "Chair",
  td.type "Type of department",
  COUNT(t.id) AS "Amount of teachers"
FROM Chair c
JOIN StructureDepartment sd ON c.structureDepartment = sd.id
JOIN TypeOfDepartment td ON sd.type = td.id
LEFT JOIN Teacher t ON c.id = t.chair
GROUP BY c.name, td.type;

```

-----###Запит №18###-----

----Відобразити контактні дані всіх кафедр певного структурного підрозділу

```

SELECT StructureDepartment.name AS "Structure Department",
Chair.name AS "Chair", Chair.phoneNumber AS "Phone",
Chair.website AS "Website"
FROM StructureDepartment
JOIN Chair ON Chair.structureDepartment = StructureDepartment.id
WHERE StructureDepartment.id = 18;

```

-----###Запит №19###-----

----Відобразити всі вчені звання, які викладачі певної кафедри

----теоретично могли б мати

```

SELECT Teacher.lastname || ' ' || Teacher.firstname || ' ' || Teacher.patronymic AS "Teacher",
Teacher.chair AS "Chair", EducationalDegree.educationaldegree
FROM Teacher
CROSS JOIN EducationalDegree
WHERE Teacher.chair = 15;

```

-----###Запит №20###-----



----Виведення контрольних заходів та їх предметів, які  
 ----мають фіксований розмір більший за 100 академічних годин

```
SELECT
s.name AS "Subject",
COUNT(ctrl.id) AS "Amount of hours"
FROM Subject s
JOIN "Control" ctrl ON s.id = ctrl.subject
WHERE s.hours > 100
GROUP BY s.name
ORDER BY "Amount of hours" DESC;
```

----####Запит №21####----

----Відобразити студентів певного викладача, які мають оцінки з його предметів  
 ----більші за 90 балів

```
SELECT Teacher.lastname AS "Teacher", Student.secondname AS "Student",
Mark.score AS "Mark", Mark.markECTS AS "ECTS"
FROM "Control"
JOIN Student ON "Control".student = Student.id
JOIN Teacher ON "Control".teacher = Teacher.id
JOIN Mark ON "Control".mark = Mark.id
WHERE "Control".typeOfControl = (SELECT id FROM TypeOfControl WHERE typeofcontrol = 'Exam')
GROUP BY Teacher.lastname, Student.secondname, Mark.score, Mark.markECTS
HAVING (Mark.score) >= 90;
```

## ДОДАТОК 3 ТЕКСТИ ПРОГРАМНОГО КОДУ СТВОРЕННЯ ІНДЕКСІВ

---ІНДЕКС №1---

```
CREATE INDEX idxStudentPersonalData ON Student(firstname, secondname, patronymic)
```

---ІНДЕКС №2---

```
CREATE INDEX idxTeacherPersonalData ON Teacher(firstname, lastname, patronymic);
```

---ІНДЕКС №3---

```
CREATE INDEX idxStudentAcademicGroup ON Student(academicgroup);
```

```
CREATE INDEX idxAcademicGroup ON AcademicGroup(id);
```

---ІНДЕКС №4---

```
CREATE INDEX idxChairDepartment ON Chair(structuredepartment);
```

```
CREATE INDEX idxStructureDepartment ON StructureDepartment(id);
```

---ІНДЕКС №5---

```
CREATE INDEX idxTeacherChair ON Teacher(chair);
```

```
CREATE INDEX idxChairID ON Chair(id);
```

```
CREATE INDEX idxControlID ON "Control"(id);
```

```
CREATE INDEX idxControlTeacher ON "Control"(teacher);
```

```
CREATE INDEX idxControlSubject ON "Control"(subject);
```

```
CREATE INDEX idxControlTypeOfControl ON "Control"(typeofcontrol);
```

```
CREATE INDEX idxControlMark ON "Control"(mark);
```

```
CREATE INDEX idxControlStudent ON "Control"(student);
```

```
CREATE INDEX idxStudentID ON Student(id);
```

```
CREATE INDEX idxTeacherID ON Teacher(id);
```

```
CREATE INDEX idxTypeOfControl ON TypeOfControl(id);
```

```
CREATE INDEX idxMarkID ON Mark(id);
```

```
CREATE INDEX idxSubjectID ON Subject(id);
```

## ДОДАТОК И    ТЕКСТИ ПРОГРАМНОГО КОДУ ІМПОРТУ ДАНИХ

--Імпортуємо дані про типи структурних підрозділів у відповідну таблицю

--з CSV файлу та виконуємо вибірку даних для перевірки успішності імпорту

\copy TypeOfDepartment(id, type)

FROM            'C:\Users\nazar\OneDrive\Робочий            стіл\ІІІ-22\ІІІ            семестр\Курсова            робота.            Бази  
даних\Скрипти\DataToImport\typeofdepartment.csv'  
WITH DELIMITER ',';

SELECT \* FROM TypeOfDepartment;

\copy EducationalDegree(id, educationalDegree)

FROM            'C:\Users\nazar\OneDrive\Робочий            стіл\ІІІ-22\ІІІ            семестр\Курсова            робота.            Бази  
даних\Скрипти\DataToImport\educationaldegree.csv'  
WITH DELIMITER ',';

SELECT \* FROM EducationalDegree;

\copy TypeOfControl(id, typeOfControl)

FROM            'C:\Users\nazar\OneDrive\Робочий            стіл\ІІІ-22\ІІІ            семестр\Курсова            робота.            Бази  
даних\Скрипти\DataToImport\typeofcontrol.csv'  
WITH DELIMITER ',';

SELECT \* FROM TypeOfControl;

\copy TypeOfSubject(id, type)

FROM            'C:\Users\nazar\OneDrive\Робочий            стіл\ІІІ-22\ІІІ            семестр\Курсова            робота.            Бази  
даних\Скрипти\DataToImport\typeofsubject.csv'  
WITH DELIMITER ',';

SELECT \* FROM TypeOfSubject;

\copy StructureDepartment(id, name, phoneNumber, email, website, type)

FROM            'C:\Users\nazar\OneDrive\Робочий            стіл\ІІІ-22\ІІІ            семестр\Курсова            робота.            Бази  
даних\Скрипти\DataToImport\structuredepartment.csv'  
WITH DELIMITER ',';

SELECT \* FROM StructureDepartment;

\copy Chair(id, name, phoneNumber, email, website, type)

FROM 'C:\Users\nazar\OneDrive\Робочий стіл\ІІІ-22\ІІІ семестр\Курсова робота. Бази даних\Скрипти\DataToImport\chair.csv'  
WITH DELIMITER ',';

SELECT \* FROM Chair;

\copy Teacher(id, firstname, lastname, patronymic, beginningworking, chair, educationaldegree)

```

FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\teacher.csv'
WITH DELIMITER ',';

SELECT * FROM Teacher;

\copy AcademicGroup(id, name, chair, academiccurator)
FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\academicgroup.csv'
WITH DELIMITER ',';

SELECT * FROM AcademicGroup;

\copy Student(firstname, secondname, patronymic, dateofentry, academicgroup)
FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\student.csv'
WITH DELIMITER ',';

SELECT * FROM Student;

\copy Subject(id, name, hours, typeofsubject)
FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\subject.csv'
WITH DELIMITER ',';

SELECT * FROM Subject;

\copy Mark(id, score, isResultingMark)
FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\resultingmarks.csv'
WITH DELIMITER ',';

SELECT * FROM Mark WHERE isResultingMark = TRUE ORDER BY score DESC;

\copy Mark(score, isResultingMark)
FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\nonresultingmarks.csv'
WITH DELIMITER ',';

\copy "Control"(attestation, dateofcontrol, typeofcontrol, subject, student, teacher)
FROM      'C:\Users\nazar\OneDrive\Робочий
даних\Скрипти\DataToImport\calendarcontrol.csv'
WITH DELIMITER ',';

SELECT * FROM "Control";

```

```
\copy "Control"(dateofcontrol, typeofcontrol, subject, student, teacher, mark)
```

```
FROM          'C:\Users\nazar\OneDrive\Робочий      стіл\ІІІ-22\ІІІ      семестр\Курсова      робота.      Бази  
даних\Скрипти\DataToImport\control.csv'
```

```
WITH DELIMITER ',';
```

```
SELECT * FROM "Control";
```

## ДОДАТОК К СТВОРЕННЯ КОРИСТУВАЧІВ БАЗИ ДАНИХ

```
CREATE ROLE controlReviewerRole;  
GRANT SELECT ON "Control", Teacher,  
Student, TypeOfControl, TypeOfSubject TO controlReviewerRole;
```

```
CREATE ROLE controlEditorRole;  
GRANT INSERT, SELECT, UPDATE, DELETE ON "Control" TO controlEditorRole;
```

```
CREATE ROLE studentManagement;  
GRANT INSERT, DELETE, UPDATE, SELECT ON Student, AcademicGroup TO studentManagement;
```

```
CREATE ROLE subjectManagement;  
GRANT INSERT, DELETE, UPDATE, SELECT ON Subject TO subjectManagement;
```

```
CREATE USER student WITH PASSWORD 'abc';  
CREATE USER cathedra WITH PASSWORD 'abc';  
CREATE USER university WITH PASSWORD 'abc';  
CREATE USER teacher WITH PASSWORD 'abc';
```

```
GRANT controlReviewerRole TO student;
```

```
GRANT controlReviewerRole TO Teacher;  
GRANT controlEditorRole TO Teacher;
```

```
GRANT studentManagement TO cathedra;  
GRANT subjectManagement TO cathedra;
```

```
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO university;
```