МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

“Харківський авіаційний інститут”

Факультет програмної інженерії та бізнесу

Кафедра інженерії програмного забезпечення

**КУРСОВА РОБОТА**

з дисципліни « Бази даних (КР)»

на тему: «ОБЛІК УСПІШНОСТІ СТУДЕНТІВ У ПОТОЧНІЙ СЕСІЇ**»**

Виконала: студентка 3 курсу групи № 633П

Спеціальності 121 «Інженерія програмного забезпечення»

Виконала: Коляндра Ю. М.

(прізвище та ініціали)

Прийняв: аспірант на кафедрі 603.

Носиков О. С.

(посада, науковий ступінь, прізвище та ініціали)

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

Кількість балів: \_\_\_\_\_\_\_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_\_\_\_\_\_

Харків – 2023

# **ПЕРЕЛІК СКОРОЧЕНЬ**

БД – База даних

ПЗ – програмне забезпечення

ПІБ – "Прізвище, Ім'я, по Батькові"

MySQL – "My Structured Query Language"

SQL – "Structured Query Language"

# **РЕФЕРАТ УКРАЇНСЬКОЮ МОВОЮ**

Сьогодні база даних є гарним інструментом для зберігання та організації даних. Її можна використовувати всюди, зокрема і у навчанні.

**Мета роботи та її етапи**

Метою цієї курсової роботи було розроблення бази даних для обліку успішності студентів з метою забезпечення зберігання, організації та зручного доступу до корисної інформації: про студентів, результати їхніх екзаменів, курси, факультети та групи університету.

Для досягнення цієї мети було проведено наступні етапи роботи:

* розроблення логічної моделі бази даних: визначення таблиць, ключів та зв'язків між ними для логічної організації даних.
* розроблення фізичної моделі бази даних: визначення структури таблиць, їх полів та типів даних для роботи з сервером MySQL.
* визначення SQL запитів та обсягу даних: розроблення SQL-запитів, оцінення обсягу даних у таблицях та трудомісткості виконання запитів.
* денормалізація фізичної моделі даних: проведення денормалізації для оптимізації виконання запитів.
* оптимізація та індексування: внесення індексів та оптимізація полів таблиць для підвищення швидкодії запитів.
* розроблення збережених процедур та функцій для часто використовуваних запитів.

Розроблена база даних має на меті покращити роботу викладачів та студентів з оцінками, що покращить ефективності завершення навчального семестру. Студенти та викладачі зможуть бачити загальну картину успішності за семестр, що вплине на здобувачів освіти, бо буде стимул покращити оцінку. Також студенти можуть проаналізувати дані і визначити на які предмети потрібно звернути увагу, якщо такі є.

Отже, розроблена база даних є потужним інструментом для обліку успішності студентів, допомагаючи покращити якість оцінювання та навчання в університеті, а також оптимізувати ці процеси.

# **РЕФЕРАТ АНГЛІЙСЬКОЮ МОВОЮ**

Today, a database serves as a valuable tool for storing and organizing data, finding applications across various fields, including education.

**The aim of coursework and its stages**

The purpose of this course work was to develop a database for student performance records in order to provide storage, organization and convenient access to useful information: about students, their exam results, courses, faculties and groups of the university.

To achieve this goal, the following stages of work were carried out:

* development of a logical database model: definition of tables, keys and relationships between them for the logical organization of data.
* development of the physical model of the database: determination of the structure of the tables, their fields and data types for working with the MySQL server.
* definition of SQL queries and the volume of data: development of SQL queries, evaluation of the volume of data in the tables and complexity of query execution.
* denormalization of the physical data model: denormalization to optimize the execution of requests.
* optimization and indexing: introduction of indexes and optimization of table fields to increase the speed of queries.
* development of stored procedures and functions for frequently used queries.

The developed database aims to improve the work of teachers and students with grades, which will improve the efficiency of completing the academic semester. Students and teachers will be able to see the overall picture of success for the semester, which will affect students, because there will be an incentive to improve the grade. Also, students can analyze the data and determine which subjects, if any, need attention.

Therefore, the developed database is a powerful tool for recording student performance, helping to improve the quality of evaluation and teaching at the university, as well as to optimize these processes.

**ЗМІСТ**

[ПЕРЕЛІК СКОРОЧЕНЬ 2](#_Toc154697294)

[РЕФЕРАТ УКРАЇНСЬКОЮ МОВОЮ 3](#_Toc154697295)

[РЕФЕРАТ АНГЛІЙСЬКОЮ МОВОЮ 4](#_Toc154697296)

[ВСТУП 6](#_Toc154697297)

[ПРИЗНАЧЕННЯ БАЗИ ДАНИХ 7](#_Toc154697298)

[ПОБУДОВА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ ДАНИХ 8](#_Toc154697299)

[ПОБУДОВА ЛОГІЧНОЇ МОДЕЛІ ДАНИХ 10](#_Toc154697300)

[ПОБУДОВА ФІЗИЧНОЇ МОДЕЛІ ДАНИХ 12](#_Toc154697301)

[ДЕНОРМАЛІЗАЦІЯ ФІЗИЧНОЇ МОДЕЛІ ДАНИХ 18](#_Toc154697302)

[РЕФАКТОРИНГ БАЗИ ДАНИХ 21](#_Toc154697303)

[ПОБУДОВА ТРИГЕРІВ БАЗИ ДАНИХ 25](#_Toc154697304)

[АВТОМАТИЗАЦІЯ ЗАПИТІВ ДО БАЗИ ДАНИХ 29](#_Toc154697305)

[АВТОМАТИЗАЦІЯ ОБРОБЛЕННЯ ПОДІЙ У БАЗІ ДАНИХ 31](#_Toc154697306)

[ВИСНОВОК 33](#_Toc154697307)

[ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 34](#_Toc154697308)

[ДОДАТОК 35](#_Toc154697309)

# **ВСТУП**

Для ефективного відстеження успішності студентів критично важливо мати повну інформацію про всі їхні оцінки, особливо за складеними екзаменами. Розробка бази даних для обліку цих даних є ключовим компонентом, щоб забезпечити оперативність, точність та зручність ведення обліку успішності.

Предметом даної курсової роботи є створення бази даних для відстеження успішності студентів. Зберігання інформації про складені екзамени та отримані оцінки є ключовим для поліпшення навчальних досягнень студентів і спрощення роботи викладачів.

Робота спрямована на розроблення структурованої бази даних, яка дозволить зберігати, оновлювати та вибирати необхідну інформацію, враховуючи взаємозв'язки між об'єктами. Реалізація такої бази даних є важливим інструментом для підвищення якості освіти в університеті.

У роботі будуть розглянуті основні складові бази даних для відстеження успішності студентів, їх взаємозв'язки та методи оптимізації процесів зберігання та обробки інформації з метою покращення навчальних результатів студентів.

# **ПРИЗНАЧЕННЯ БАЗИ ДАНИХ**

Створення бази даних для обліку успішності студентів у поточній сесії може значно полегшити відстеження успішності студентів, зробити зручним перегляд та редагування інформації про номер групи студента, факультет групи, ПІБ викладача і лектора, номер залікової книжки студента, найменування дисциплін, оцінки за іспити та дату їх складання.

**Головні задачі, які буде вирішувати програма**

Невід’ємною складовою розробленої системи – веб-сервісу з обліку успішності студентів у поточній сесії є функції адміністратора. Усі важливі питання, що пов’язані з роботою з даними, а саме, редагування інформації про номер групи студента, факультет групи, ПІБ викладача і лектора, номер залікової книжки студента, найменування дисциплін, оцінки за іспити та дату їх складання, вирішуються через адміністратора.

При роботі з системою адміністратор має мати можливість переглядати дані всіх таблиць, за необхідності додавати або видаляти певні дані з цих таблиць.

Оператор системи – це користувач, якій веде облік факультетів, предметів, номерів груп, ПІБ про студентів та лекторів, номерів залікових книжок, оцінок та дат складання.

# **ПОБУДОВА КОНЦЕПТУАЛЬНОЇ МОДЕЛІ ДАНИХ**

Концептуальна модель зображена на рисунку 1.

**Головні сутності предметної області та їх властивості**

Наведено перелік сутностей та їх властивостей:

**Сутність «Студент»**

Студент, який навчається в університеті.

Властивості:

* ПІБ студента
* Номер залікової книжки

**Сутність «Екзамен»**

Екзамен – те, що має скласти студент.

* Оцінка
* Дата складання

**Сутність «Викладач»**

Викладач – приймає екзамен

Властивості:

* ПІБ викладача

**Сутність «Дисципліна»**

Дисципліна – предмет, який вивчає студент і з якого складає екзамен.

Властивості:

* Назва дисципліни

**Сутність «Група»**

Група, до якої відноситься студент

Властивості:

* Номер групи

**Сутність «Факультет»**

Факультет, до якого належить група

Властивості:

* Назва факультету

**Зв’язки між сутностями та їх кардинальність**

**Зв’язок «Група» (1) --- (багато) «Студент»**

* Одна група має багато студентів.
* Один студент має одну групу.

**Зв’язок «Факультет»(1) – (багато) «Група»**

* Один факультет має багато груп.
* Група належить до одного факультету.

**Зв’язок «Студент»(багато) – (багато) «Екзамен»**

* Один студент складає багато екзаменів.
* Багато студентів складають один екзамен.

**Зв’язок «Дисципліна»(1) – (багато) «Екзамен»**

* Одна дисципліна може мати багато екзаменів.
* Один екзамен може відноситися лише до однієї дисципліни.

**Зв’язок «Дисципліна» (багато) – (1) «Викладач»**

* Один викладач може вести багато дисциплін.
* Дисципліну може вести лише один викладач.

**Зв’язок «Викладач»(1) –(багато) «Екзамен»**

* Один викладач може приймати декілька екзаменів.

Зображення, що містить схема, ескіз, малюнок, текст

Автоматично згенерований опис

Рисунок 1 – Концептуальна модель даних

# **ПОБУДОВА ЛОГІЧНОЇ МОДЕЛІ ДАНИХ**

Побудована логічна модель БД зображена на рисунку 2.

**Склад відношень, що відповідають сутностям та їх атрибути**

**Сутність «Студент»**

* КодСтудента: Цілі числа (унікальний ідентифікатор студента)
* ПІБ: Рядок символів (прізвище, ім’я, по-батькові студента)
* НомерЗаліковоїКнижки: Цілі числа (номер залікової книжки студента)
* КодГрупи: Цілі числа (посилання на унікальний ідентифікатор групи)

**Сутність «Екзамен»**

* КодЕкзамену: Цілі числа (унікальний ідентифікатор екзамену)
* КодДисципліни: Цілі числа (посилання на унікальний ідентифікатор дисципліни, екзамен з якого складає студент)
* КодСтудента: Цілі числа (посилання на унікальний ідентифікатор студента, який складає екзамен)
* ДатаСкладання: Дата (дата складання екзамену)
* Оцінка: Дійсне число (оцінка за екзамен)

**Сутність «Викладач»**

* КодВикладача: Цілі числа (унікальний ідентифікатор викладача)
* ПІБ: Рядок символів (прізвище, ім’я, по-батькові викладача)

**Сутність «Дисципліна»**

* КодДисципліни: Цілі числа (унікальний ідентифікатор дисципліни)
* КодВикладача: Цілі числа (посилання на унікальний ідентифікатор викладача)
* НазваДисципліни: Рядок символів (назва дисципліни)

**Сутність «Група»**

* КодГрупи: Цілі числа (унікальний ідентифікатор групи)
* НомерГрупи: Рядок символів (номер групи)
* КодФакультету: Цілі числа (посилання на унікальний ідентифікатор факультету)

**Сутність «Факультет»**

* КодФакультету: Цілі числа (унікальний ідентифікатор факультету)
* НазваФакультету: Рядок символів (назва факультету)

**Нормалізація відношень та зв’язків логічної моделі даних**

**Перша нормальна форма**

Усі відношення логічної моделі даних знаходяться в 1НФ. Всі значення атрибутів атомарні. Кожен атрибут відношення містить лише одне значення, і не є множиною значень або повторюваних груп.

**Друга нормальна форма**

Усі відношення логічної моделі даних знаходяться в 2НФ, бо модель даних вже перебуває в 1НФ. Також кожний атрибут у таблиці, що не є первинним ключем, повністю залежить від ключа у цілому.

**Третя нормальна форма**

Усі відношення логічної моделі даних знаходяться в 3НФ, адже модель даних вже перебуває в 2НФ. Також не ключовими атрибутами відношення не мають функціональних зв’язків.

Зображення, що містить текст, знімок екрана, схема, ряд

Автоматично згенерований опис

Рисунок 2 – Логічна модель даних

# **ПОБУДОВА ФІЗИЧНОЇ МОДЕЛІ ДАНИХ**

**Визначення складу таблиць, що будуть відповідати відношенням**

Таблиця 1 – Сутність «Студент»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Опис* |
| КодСтудента | Ціле число | Первинний ключ (1…10000) |
| ПІБ | Рядок | Макс. довжина рядку = 255 |
| НомерЗаліковоїКнижки | Ціле число | Макс. довжина рядку = 20 |
| КодГрупи | Ціле число | Зовнішній ключ (1…1000) |

Таблиця 2 – Сутність «Екзамен»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Опис* |
| КодЕкзамену | Ціле число | Первинний ключ (1…10000) |
| КодДисципліни | Ціле число | Зовнішній ключ (1…1000) |
| КодСтудента | Ціле число | Зовнішній ключ (1…1000) |
| ДатаСкладання | Дата | Формат дати РРРР-ММ-ДД |
| Оцінка | Дійсне число | Два числа після коми |

Таблиця 3 – Сутність «Викладач»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Опис* |
| КодВикладача | Ціле число | Первинний ключ (1…10000) |
| ПІБ | Рядок | Макс. довжина рядку = 255 |

Таблиця 4 – Сутність «Дисципліна»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Опис* |
| КодДисципліни | Ціле число | Первинний ключ (1…100) |
| КодВикладача | Ціле число | Зовнішній ключ (1…1000) |
| НазваДисципліни | Рядок | Макс. довжина рядку = 255 |

Таблиця 5 – Сутність «Група»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Опис* |
| КодГрупи | Ціле число | Первинний ключ (1…1000) |
| НомерГрупи | Ціле число | Макс. довжина рядку = 50 |
| КодФакультету | Ціле число | Зовнішній ключ (1…1000) |

Таблиця 6 – Сутність «Факультет»

|  |  |  |
| --- | --- | --- |
| *Атрибути* | *Тип* | *Опис* |
| КодФакультету | Ціле число | Первинний ключ (1…100 000) |
| НазваФакультету | Рядок | Макс. довжина рядку = 255 |

**Визначення стовпців таблиць та їх типи даних**

Таблиця 7 – Таблиця «Student»

|  |  |  |
| --- | --- | --- |
| *Поля* | *Тип* | *Опис* |
| student\_id | int | Первинний ключ (1…1000000), 4 байт |
| username | varchar(255) | Макс. довжина рядку = 255, 256 байт |
| grades\_record\_number | int | 100000…1000000, 4 байт |
| group\_id | smallint | Зовнішній ключ (1…1000), 2 байти |

Таблиця 8 – Таблиця «Exam»

|  |  |  |
| --- | --- | --- |
| *Поля* | *Тип* | *Опис* |
| exam\_id | int | Первинний ключ (1…1000000) , 4 байт |
| course\_id | smallint | Зовнішній ключ (1…1000), 2 байти |
| student\_id | int | Зовнішній ключ (1…1000000), 4 байти |
| date | date | Формат дати РРРР-ММ-ДД  (до 2050-01-01), 3 байти |
| grade | decimal | Дві цифри після коми, 4 байти |

Таблиця 9 – Таблиця «Professor»

|  |  |  |
| --- | --- | --- |
| *Поля* | *Тип* | *Опис* |
| professor\_id | int | Первинний ключ (1…1000000) , 4 байт |
| prof\_username | varchar(255) | Макс. довжина рядку = 255, 256 байт |

Таблиця 10 – Таблиця «Course»

|  |  |  |
| --- | --- | --- |
| *Поля* | *Тип* | *Опис* |
| course\_id | smallint | Первинний ключ (1…1000), 2 байти |
| coursename | varchar(255) | Макс. довжина рядку = 255, 256 байт |
| professor\_id | smallint | Зовнішній ключ (1…1000), 2 байти |

Таблиця 11 – Таблиця «Group»

|  |  |  |
| --- | --- | --- |
| *Поля* | *Тип* | *Опис* |
| group\_id | smallint | Первинний ключ (1…1000) 2 байти |
| group\_number | smallint | 100…999) , 2 байти |
| faculty\_id | tinyint | Зовнішній ключ (1…100), 1 байт |

Таблиця 12 – Сутність «Faculty»

|  |  |  |
| --- | --- | --- |
| *Поля* | *Тип* | *Опис* |
| faculty\_id | tinyint | Первинний ключ (1…100), 1 байт |
| faculty\_name | varchar(255) | Макс. довжина рядку = 255, 256 байт |

**Доведення, що сервер MySQL можливо використати як носій даних**

Сервер MySQL може використовуватися як носій даних, бо:

* зберігання даних на MySQL дозволяє зберігати їх навіть після перезавантаження сервера;
* може забезпечити швидкий доступ до даних, оскільки може використовувати різні оптимізації для швидшого читання та запису інформації;
* може бути простішим у використанні та управлінні, особливо для менших систем або там, де не потрібна розподіленість даних;
* надає кроссплатформенну базу даних: (працює на Linux, Windows та інших операційних системах);
* пропонує користувачам найвищий рівень безпеки, має вбудовані інструменти безпеки, які підтримують управління користувачами та їх привілеями.

Побудована фізична модель БД зображена на рисунку 3.

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок 3 – Фізична модель даних

**Визначення мовою SQL запитів до бази даних**

Було визначено такі запити до бази даних обліку успішності студентів:

Отримати список студентів разом із групою, назвою дисципліни, ПІБ лектора, датою здачі та оцінкою:

SELECT

s.student\_id,

s.username AS student\_name,

g.group\_number,

c.coursename AS course\_name,

p.prof\_username AS professor\_name,

e.date,

e.grade

FROM

Student s

JOIN

Exam e ON s.student\_id = e.student\_id

JOIN

Course c ON e.professor\_id = c.professor\_id

JOIN

Group g ON s.group\_id = g.group\_id

JOIN

Professor p ON c.professor\_id = p.professor\_id;

Вибрати всіх студентів певної групи та їхні оцінки за конкретний курс:

SELECT

s.student\_id,

s.username AS student\_name,

e.grade,

e.date

FROM

Student s

JOIN

Exam e ON s.student\_id = e.student\_id

JOIN

Course c ON e.professor\_id = c.professor\_id

WHERE

s.group\_id = <group\_id>

AND c.course\_id = <course\_id>;

Отримати список студентів, які навчаються на певному факультеті:

SELECT

s.student\_id,

s.username AS student\_name,

s.grades\_record\_number,

s.group\_id

FROM

Student s

JOIN

Group g ON s.group\_id = g.group\_id

JOIN

Faculty f ON g.faculty\_id = f.faculty\_id

WHERE

f.faculty\_id = <faculty\_id>;

**Визначення обсягу даних у таблицях та трудомісткості виконання запитів**

Розроблена фізична модель даних складається з 6 таблиць, а саме: Student, Exam, Professor, Group, Course, Faculty. Кожний стовпчик таблиць відповідає атрибутам відповідних відношень. При розробці фізичної моделі даних для MySQL 8.0 було враховано: типи даних, первинні та зовнішні ключі та обмеження, щоб забезпечити ефективну роботу бази даних.

Таблиця 13 – Довжини стовпчиків та приблизна кількість записів

|  |  |  |  |
| --- | --- | --- | --- |
| № | Таблиці | Довжина запису | Кількість записів |
| 1 | Student | 266 | 10 000 |
| 2 | Exam | 17 | 50 000 |
| 3 | Professor | 260 | 250 |
| 4 | Course | 260 | 500 |
| 5 | Group | 5 | 500 |
| 6 | Faculty | 257 | 10 |

Розрахуємо трудомісткість першого запиту з попереднього пункту:

У запиті приймає участь таблиці Student(10000), Exam(50000), Course(500), Group(500). У дужках зазначена кількість записів у таблицях.

Загальні кількість записів які обробляються запитом: 10000\*50000\*500\*500=125\*1012. Загальна довжина дорівнює сумі довжин записів цих табличок, тобто 266+17+260+5=543. Загальний обсяг даних 125\*1012\*543=67875\*1012 байт.

Розрахуємо трудомісткість другого запиту з попереднього пункту:

У запиті приймають участь таблиці Student(10000), Exam(50000), Course(500).

Тобто загальні кількість записів які обробляються запитом: 10000\*50000\*500=25\*1010. Загальна довжина дорівнює сумі довжин записів цих табличок, тобто 266+17+260=543. Загальний обсяг даних 25\*1010\*543=13575\*1010 байт.

Розрахуємо трудомісткість третього запиту з попереднього пункту:

У запиті приймають участь таблиці Student(10000), Group(500), Faculty(10).

Тобто загальна кількість записів які обробляються запитом: 10000\*500\*10=5\*107 Загальна довжина дорівнює сумі довжин записів цих табличок, тобто 266+5+257=528. Загальний обсяг даних 5\*107\*528=264\*108 байт.

# **ДЕНОРМАЛІЗАЦІЯ ФІЗИЧНОЇ МОДЕЛІ ДАНИХ**

**Оцінка ефективністі денормалізації фізичної моделі даних**

Для денормалізації даних в таблицях та для виконання запитів «Отримати список студентів разом із групою, назвою дисципліни, ПІБ лектора, датою здачі та оцінкою» та «Вибрати всіх студентів певної групи та їхні оцінки за конкретний курс», можемо додати у таблицю Exam поля prof\_username та coursename, у таблицю Course поле prof\_username і у таблиці Group та Student поле facultyname.

Відповідно, значення таблиць зміняться (таблиця 14).

Таблиця 14 – Нові довжини стовпчиків та приблизна кількість записів

|  |  |  |  |
| --- | --- | --- | --- |
| № | Таблиці | Довжина запису | Кількість записів |
| 1 | Student | 521 | 10 000 |
| 2 | Exam | 527 | 50 000 |
| 3 | Professor | 260 | 250 |
| 4 | Course | 515 | 500 |
| 5 | Group | 260 | 500 |
| 6 | Faculty | 257 | 10 |

Загальний обсяг даних для першого запиту таким чином складе:

загальна кількість записів які обробляються запитом: 10000\*50000\*500=25\*1010

загальна довжина записів: 521+527+260=1308

загальний обсяг даних: 25\*1010\*1308=327\*1012 байт

Загальний обсяг даних для другого запиту таким чином складе:

загальна кількість записів які обробляються запитом: 10000\*50000=5\*108

загальна довжина записів: 521+527=1048

загальний обсяг даних: 5\*108\*1048=524\*109 байт

Загальний обсяг даних для третього запиту таким чином складе:

загальна кількість записів які обробляються запитом: 10000\*500=5\*106

загальна довжина записів: 521+260=781

загальний обсяг даних: 5\*106\*781=3905\*106 байт

Денормалізована модель даних наведена на рисунку 4.

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок 4 – Денормалізована модель даних

**Визначення запитів до денормалізованої фізичної моделі даних**

Змінені запити до денормалізованої моделі:

Отримати список студентів разом із групою, назвою дисципліни, ПІБ лектора, датою здачі та оцінкою:

SELECT

e.student\_id,

s.username AS student\_name,

g.group\_number,

e.course\_name,

e.prof\_username,

e.date,

e.grade

FROM

Exam e

JOIN

Student s ON e.student\_id = s.student\_id

JOIN

Group g ON s.group\_id = g.group\_id;

Вибрати всіх студентів певної групи та їхні оцінки за конкретний курс:

SELECT

e.student\_id,

s.username AS student\_name,

e.grade,

e.date

FROM

Exam e

JOIN

Student s ON e.student\_id = s.student\_id

WHERE

s.group\_id = <group\_id>

AND e.course\_name = '<course\_name>';

Отримати список студентів, які навчаються на певному факультеті:

SELECT

s.student\_id,

s.username AS student\_name,

s.grades\_record\_number,

s.group\_id

FROM

Student s

JOIN

Group g ON s.group\_id = g.group\_id

WHERE

g.faculty\_name = '<faculty\_name>';

# **РЕФАКТОРИНГ БАЗИ ДАНИХ**

**Проведення кластеризації таблиць фізичної моделі даних**

Кластеризація таблиць у фізичній моделі даних в базі даних SQL часто визначається способом зберігання даних на диску. Зазвичай використовується кластеризація за первинним ключем таблиці.

У моїй базі даних ключі вже було кластеризовано під час генерації коду фізичної моделі даних:

CREATE TABLE [Course] (

[course\_id] int NOT NULL,

[coursename] varchar(255) NULL,

[professor\_id] int NULL,

CONSTRAINT [\_copy\_5] PRIMARY KEY CLUSTERED ([course\_id])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

)

GO

CREATE TABLE [Exam] (

[exam\_id] int NOT NULL,

[student\_id] int NULL,

[professor\_id] int NULL,

[date] date NULL,

[grade] real NULL,

CONSTRAINT [\_copy\_1] PRIMARY KEY CLUSTERED ([exam\_id])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

)

GO

CREATE TABLE [Faculty] (

[faculty\_id] int NOT NULL,

[facultyname] varchar(255) NULL,

CONSTRAINT [\_copy\_3] PRIMARY KEY CLUSTERED ([faculty\_id])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

)

GO

CREATE TABLE [Group] (

[group\_id] int NOT NULL,

[group\_number] int NULL,

[faculty\_id] int NULL,

CONSTRAINT [\_copy\_4] PRIMARY KEY CLUSTERED ([group\_id])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

)

GO

CREATE TABLE [Proffesor] (

[professor\_id] int NOT NULL,

[username] varchar(255) NULL,

CONSTRAINT [\_copy\_6] PRIMARY KEY CLUSTERED ([professor\_id])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

)

GO

CREATE TABLE [Student] (

[student\_id] int NOT NULL,

[username] varchar(255) NULL,

[grades\_record\_number] int NULL,

[group\_id] int NULL,

CONSTRAINT [\_copy\_2] PRIMARY KEY CLUSTERED ([student\_id])

WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON)

)

GO

ALTER TABLE [Course] ADD CONSTRAINT [fk\_course\_proffesor] FOREIGN KEY ([professor\_id]) REFERENCES [Proffesor] ([professor\_id])

GO

ALTER TABLE [Exam] ADD CONSTRAINT [fk\_exam\_professor] FOREIGN KEY ([professor\_id]) REFERENCES [Proffesor] ([professor\_id])

GO

ALTER TABLE [Exam] ADD CONSTRAINT [fk\_exam\_student] FOREIGN KEY ([student\_id]) REFERENCES [Student] ([student\_id])

GO

ALTER TABLE [Group] ADD CONSTRAINT [fk\_group\_faculty] FOREIGN KEY ([faculty\_id]) REFERENCES [Faculty] ([faculty\_id])

GO

ALTER TABLE [Student] ADD CONSTRAINT [fk\_student\_group] FOREIGN KEY ([group\_id]) REFERENCES [Group] ([group\_id])

GO

**Внесення представлення у фізичну модель даних**

Представлення для відображення інформації про студентів та екзамени, які вони склали (з оцінкою, датою здачі та ПІБ викладача):

CREATE VIEW track\_view AS

SELECT

student.username,

student.grades\_record\_number,

student.group\_id,

exam.exam\_id,

exam.coursename,

exam.date,

exam.grade,

exam.prof\_username

FROM

student

INNER JOIN

exam ON student.student\_id = exam.student\_id;

Зображення, що містить текст, Шрифт, число, ряд

Автоматично згенерований опис

Рисунок 5 – Створене представлення

**Індексування полів таблиць.**

1. Створення індексу для стовпця `date` у таблиці `Exam`:

CREATE INDEX idx\_exam\_date ON Exam(date);

1. Створення складного індексу для стовпців `username` та `group\_id` у таблиці `Student`:

CREATE INDEX idx\_student\_group\_id ON Student(student\_id, group\_id);

1. Створення індексу для стовпця `facultyname` у таблиці `Faculty`:

CREATE INDEX idx\_facultyname ON Faculty(facultyname);

1. Створення індексу для стовпця `prof\_username` у таблиці `Professor`:

CREATE INDEX idx\_prof\_username ON Professor(prof\_username);

**Побудова модифікованої фізичної моделі даних**

Модифікована модель даних зображена на рисунку 6.

Зображення, що містить текст, знімок екрана, схема, Паралель

Автоматично згенерований опис

Рисунок 6 – Модифікована модель даних

# **ПОБУДОВА ТРИГЕРІВ БАЗИ ДАНИХ**

Тригер trg\_prevent\_delete\_group\_with\_students викликається **перед видаленням групи** з таблиці group. Він перевіряє чи є запис про групу у таблиці student і видає помилку і повідомлення про це, якщо запис є (це означає, що групу видалити з бази даних неможливо, якщо є дані про те, що у ній є студенти).

DELIMITER ;;

CREATE TRIGGER trg\_prevent\_delete\_group\_with\_students

BEFORE DELETE ON `group`

FOR EACH ROW

BEGIN

IF EXISTS (SELECT 1 FROM student WHERE group\_id = OLD.group\_id) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'У групі є студенти.';

END IF;

END;;

DELIMITER ;

Тригер tr\_insert\_group\_facultyname викликається **перед додаванням нової групи** у таблицю group. Він додає назву факультету, до якого відноситься група у колонку ‘facultyname’, тому під час додавання групи можна тільки ввести назву групи, id та до якого факультету вона відноситься. Це спрощує додавання нових груп.

DELIMITER ;;

CREATE TRIGGER tr\_insert\_group\_facultyname

BEFORE INSERT ON `group`

FOR EACH ROW

BEGIN

SET NEW.facultyname = (SELECT facultyname FROM faculty WHERE faculty.faculty\_id = NEW.faculty\_id);

END;;

DELIMITER ;

Тригер tr\_insert\_professorname викликається **перед додаванням нового курсу** у таблицю coutse. Він додає ПІБ викладача, який веде курс у колонку ‘prof\_username’. Це спрощує додавання нових курсів, адже потрібно ввести менше даних.

DELIMITER ;;

CREATE TRIGGER tr\_insert\_professorname

BEFORE INSERT ON `group`

FOR EACH ROW

BEGIN

SET NEW.prof\_username = (SELECT prof\_username FROM professor WHERE professor.professor\_id = NEW.professor\_id);END;;

DELIMITER ;

Тригер trg\_update\_course\_info\_in\_exam викликається **перед додаванням нового запису** у таблицю exam. Він додає ПІБ викладача, який веде курс у колонку ‘prof\_username’ і назву курсу, екзамен з якого складає студент у колонку ‘coursename’. Це спрощує додавання нових записів, адже потрібно ввести менше даних.

DELIMITER ;;

CREATE TRIGGER trg\_update\_course\_info\_in\_exam

BEFORE INSERT ON `exam`

FOR EACH ROW

BEGIN

SET NEW.coursename = (SELECT coursename FROM course WHERE course.course\_id = NEW.course\_id);

SET NEW.prof\_username = (SELECT prof\_username FROM course WHERE course.course\_id = NEW.course\_id);

END;;

DELIMITER ;

**Реалізація, налагодження та тестування внесених тригерів**

Тестування тригеру trg\_prevent\_delete\_group\_with\_students

Спочатку спробуємо видалити групу, у якій є студенти:

DELETE FROM `group` WHERE group\_id = 631

Зображення, що містить текст, знімок екрана, ряд, Шрифт

Автоматично згенерований опис

Рисунок 7 – Результат роботи тригеру trg\_prevent\_delete\_group\_with\_students

Також створимо нову групу щоб перевірити чи можна її видалити, якщо у ній не буде студентів:

INSERT INTO `group` (group\_number, faculty\_id)

VALUES (404, 4);

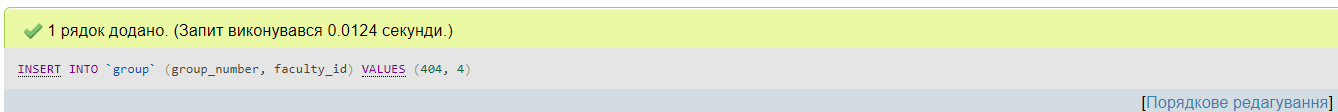


Рисунок 8 – Створення нової групи

Тепер спробуємо видалити цю групу:

DELETE FROM `group` WHERE group\_id = 404

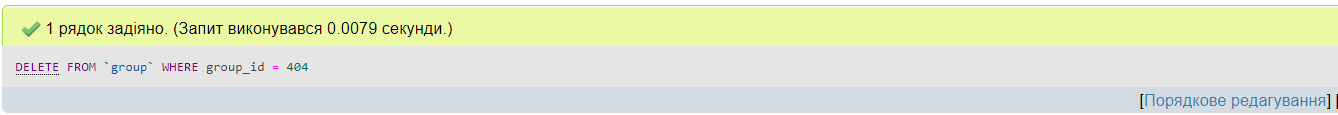


Рисунок 9 – Групу видалено успішно

Отже, тригер працює. Він не дозволяє видаляти групу, якщо у ній є студенти і дозволяє, якщо немає.

Тестування тригеру tr\_insert\_group\_facultyname

Додамо нову групу:

INSERT INTO `group` (group\_id, group\_number, faculty\_id) VALUES (611,'611п', 6);

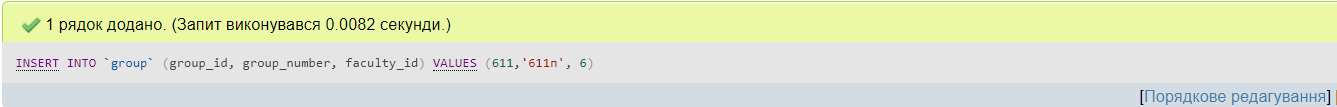


Рисунок 10 – Групу додано успішно

Бачимо, що у таблиці назва факультету автоматично додалася (рисунок 11).

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рисунок 11 – Оновлена таблиця ‘group’

Тестування тригеру tr\_insert\_group\_facultyname

Додаємо новий курс:

INSERT INTO `course` (`course\_id`, `coursename`, `professor\_id`, `prof\_username`) VALUES ('672', 'Проєктування інтерфейсу користувача', '977', NULL);

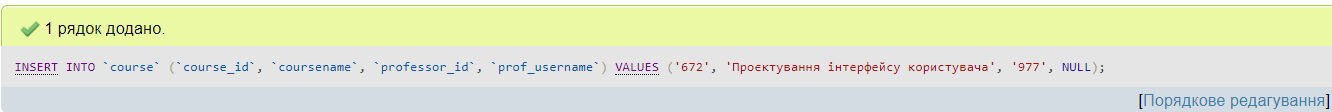


Рисунок 12 – Курс додано успішно

На рисунку 13 видно що доданий новий предмет має ПІБ викладача. Інші були додані до створення цього тригеру.

Зображення, що містить текст, Шрифт, ряд, число

Автоматично згенерований опис

Рисунок 13 – Оновлена таблиця ‘course’

Тестування тригеру trg\_update\_course\_info\_in\_exam

Додаємо новий запис:

INSERT INTO exam (exam\_id, student\_id, course\_id, date, grade)

VALUES (10005, 5, 671, '2023-12-27', 85.5);

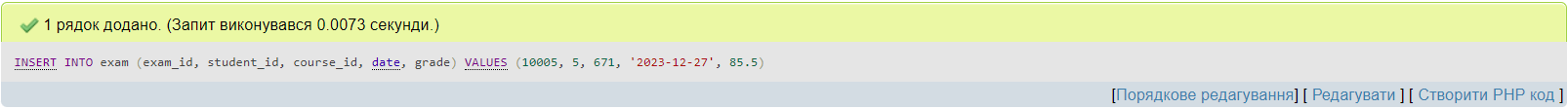


Рисунок 14 – Запис додано успішно

На рисунку 15 видно що доданий новий запис має інформацію про ПІБ викладача, який прийняв і назву курсу.

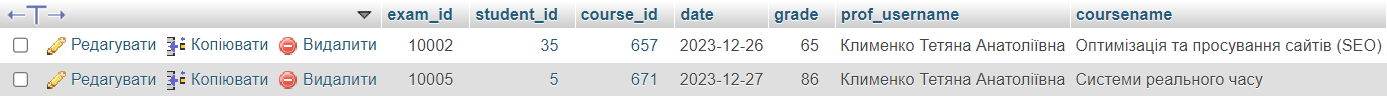


Рисунок 15 – Оновлена таблиця ‘exam’

# **АВТОМАТИЗАЦІЯ ЗАПИТІВ ДО БАЗИ ДАНИХ**

Було визначено такі запити до бази даних обліку студентів:

1. Отримати список студентів разом із групою, назвою дисципліни, ПІБ лектора, датою здачі та оцінкою:

SELECT

e.student\_id,

s.username AS student\_name,

g.group\_number,

e.course\_name,

e.profusername,

e.date,

e.grade

FROM

Exam e

JOIN

Student s ON e.student\_id = s.student\_id

JOIN

Group g ON s.group\_id = g.group\_id;

Зображення, що містить текст, Шрифт, число, знімок екрана

Автоматично згенерований опис

Рисунок 16 – Результат виконання запиту 1

1. Отримати список студентів, які навчаються на певному факультеті:

SELECT

s.student\_id,

s.username AS student\_name,

s.grades\_record\_number,

s.group\_id

FROM

Student s

JOIN

Group g ON s.group\_id = g.group\_id

WHERE

g.faculty\_name = '<faculty\_name>';

Зображення, що містить текст, Шрифт, число, знімок екрана

Автоматично згенерований опис

Рисунок 17 – Результат виконання запиту 2

Обрахунки місткості запитів наведено у розділі «РЕФАКТОРИНГ БАЗ ДАНИХ».

# **АВТОМАТИЗАЦІЯ ОБРОБЛЕННЯ ПОДІЙ У БАЗІ ДАНИХ**

Збережена процедура `AddStudent` призначена для спрощення додавання нового студента.

DELIMITER ;;

CREATE PROCEDURE AddStudent

(IN p\_Username VARCHAR(255),

IN p\_GradesRecordNumber INT,

IN p\_GroupId SMALLINT,

IN p\_FacultyName VARCHAR(255))

BEGIN

INSERT INTO student (username, grades\_record\_number, group\_id, facultyname)

VALUES (p\_Username, p\_GradesRecordNumber, p\_GroupId, p\_FacultyName);

END ;;

DELIMITER ;

Функція `AddStudent` призначена для обрахунку середнього балу студента. Вона розраховує середнє арифметичне усіх екзаменів, що склав студент. Якщо студент перездавав екзамен, то враховується останній із записів.

DELIMITER //

CREATE FUNCTION GetAverageGrade

(StudentId INT)

RETURNS DECIMAL

DETERMINISTIC

BEGIN

DECLARE AverageGrade DECIMAL;

SET AverageGrade = (

SELECT AVG(grade)

FROM (

SELECT

exam\_id,

grade,

ROW\_NUMBER() OVER (PARTITION BY coursename ORDER BY date DESC) AS RowNum

FROM exam

WHERE student\_id = StudentId

) AS OrderedGrades

WHERE RowNum = 1

);

RETURN AverageGrade;

END //

DELIMITER ;

**Тестування функцій та процедур**

Тестування процедури AddStudent

Визиваємо процедуру:

CALL AddStudent('Петренко Євген Євгенович', 12345, 631, 'Факультет програмної інженерії та бізнесу');

На рисунку 18 видно, що студента було додано.

Зображення, що містить текст, Шрифт, число, знімок екрана

Автоматично згенерований опис

Рисунок 18 – Оновлена таблиця ‘student’

Тестування функції GetAverageGrade

На рисунку 19 показано таблицю, на даних якої буде обраховуватися середній бал студента.

Зображення, що містить текст, Шрифт, число, знімок екрана

Автоматично згенерований опис

Рисунок 19 – Таблиця ‘exam’

Виклик функції:

SELECT GetAverageGrade(13) AS AverageGrade;

Результат подано на рисунку 20.

Зображення, що містить текст, Шрифт, знімок екрана

Автоматично згенерований опис

Рисунок 20 – Результат функції GetAverageGrade

# **ВИСНОВОК**

У ході цієї курсової роботи було проаналізовано та розроблено базу даних для обліку успішності студентів під час сесії з метою поліпшення управління інформацією та оптимізації процесу складання сесії.

Загалом було створено концептуальну, логічну та фізичну моделі даних. Проведено денормалізацію останньої. Також було здійснено рефакторинг бази даних. Крім цього було побудовано тригери, запити та функції і показано їх роботу. Було написано і обраховано трудомісткість запитів до бази даних.

# **ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ**

1. База даних [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/База_даних>.
2. Navicat for Education [Електронний ресурс] – Режим доступу до ресурсу: <https://www.navicat.com/en/sponsorship/education/student> .
3. Первинний ключ [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/первинний_ключ>.
4. Перша нормальна форма [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Перша_нормальна_форма>
5. Друга нормальна форма [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Друга_нормальна_форма>
6. Третя нормальна форма [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Третя_нормальна_форма>..
7. MySQL Tutorial [Електронний ресурс] – Режим доступу до ресурсу: <https://www.techonthenet.com/mysql/index.php>.
8. MySQL® Notes for Professionals book [Електронний ресурс] – Режим доступу до ресурсу: <https://books.goalkicker.com/MySQLBook/>.
9. MySQL Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.oracle.com/cd/E17952_01/index.html>.
10. Нормалізація баз даних [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/нормалізація_баз_даних>.
11. Data Types [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/data-types.html>.
12. Database refactoring [Електронний ресурс] – Режим доступу до ресурсу: <https://en.wikipedia.org/wiki/Database_refactoring>.

# **ДОДАТОК**

**Програмний код генерації БД**

-- -- phpMyAdmin SQL Dump

-- version 4.9.1

-- https://www.phpmyadmin.net/

--

-- Хост: localhost

-- Час створення: Гру 28 2023 р., 20:52

-- Версія сервера: 8.0.17

-- Версія PHP: 7.3.10

SET SQL\_MODE = "NO\_AUTO\_VALUE\_ON\_ZERO";

SET AUTOCOMMIT = 0;

START TRANSACTION;

SET time\_zone = "+00:00";

/\*!40101 SET @OLD\_CHARACTER\_SET\_CLIENT=@@CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET @OLD\_CHARACTER\_SET\_RESULTS=@@CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET @OLD\_COLLATION\_CONNECTION=@@COLLATION\_CONNECTION \*/;

/\*!40101 SET NAMES utf8mb4 \*/;

--

-- База даних: `tracking\_grades`

--

DELIMITER $$

--

-- Процедури

--

CREATE DEFINER=`root`@`localhost` PROCEDURE `AddStudent` (IN `p\_Username` VARCHAR(255), IN `p\_GradesRecordNumber` INT, IN `p\_GroupId` SMALLINT, IN `p\_FacultyName` VARCHAR(255)) BEGIN

INSERT INTO student (username, grades\_record\_number, group\_id, facultyname)

VALUES (p\_Username, p\_GradesRecordNumber, p\_GroupId, p\_FacultyName);

END$$

--

-- Функції

--

CREATE DEFINER=`root`@`localhost` FUNCTION `GetAverageGrade` (`StudentId` INT) RETURNS DECIMAL(10,0) BEGIN

DECLARE AverageGrade DECIMAL;

SET AverageGrade = (

SELECT AVG(grade)

FROM (

SELECT

exam\_id,

grade,

ROW\_NUMBER() OVER (PARTITION BY coursename ORDER BY date DESC) AS RowNum

FROM exam

WHERE student\_id = StudentId

) AS OrderedGrades

WHERE RowNum = 1

);

RETURN AverageGrade;

END$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблиці `course`

--

CREATE TABLE `course` (

`course\_id` smallint(6) NOT NULL,

`coursename` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`professor\_id` int(11) DEFAULT NULL,

`prof\_username` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Дамп даних таблиці `course`

--

INSERT INTO `course` (`course\_id`, `coursename`, `professor\_id`, `prof\_username`) VALUES

(322, 'Операційні системи', 982, 'Губка Сергій Олексійович'),

(323, 'Техн.виг-я. зас-в вим. техн-и', 983, 'Заболотний Олександр Віталійович'),

(459, 'Прийняття рішень при багатокритеріальному оцінюванні', 980, 'Шпілінська Ольга Леонідівна'),

(460, 'Проектування технологічних процесів та автоматизовані методи виробництва композитних конструкцій', 981, 'Івановська Ольга Володимирівна'),

(544, 'Цифрова схемотехніка', 978, 'Попов Анатолій Владиславович'),

(545, 'Smart Vision', 978, 'Попов Анатолій Владиславович'),

(547, 'Окремі розділи фізики', 979, 'Клименко Ігор Андрійович'),

(653, 'Програмування мовою Python', 977, 'Лучшева Оксана Вадимівна'),

(655, 'Обчислювальна техніка та програмування', 977, 'Лучшева Оксана Вадимівна'),

(656, 'Конструювання ПЗ', 974, 'Дегтярьова Тетяна Григорівна'),

(657, 'Оптимізація та просування сайтів (SEO)', 976, 'Клименко Тетяна Анатоліївна'),

(658, 'Стандартизація і сертифікація продукції та послуг', 979, 'Клименко Ігор Андрійович'),

(659, 'Web-програмування мовою PHP', 975, 'Вдовітченко Олександр Валерійович'),

(670, 'Системне програмування', 975, 'Вдовітченко Олександр Валерійович'),

(671, 'Системи реального часу', 976, 'Клименко Тетяна Анатоліївна'),

(672, 'Проєктування інтерфейсу користувача', 977, 'Лучшева Оксана Вадимівна'),

(745, 'Формування системного наукового світогляду', 972, 'Качуров Євгеній Васильович'),

(746, 'Розвиток комунікацій', 973, 'Жарікова Світлана Борисівна'),

(747, 'Загальна психологія', 973, 'Жарікова Світлана Борисівна'),

(748, 'Організаційна психологія', 973, 'Жарікова Світлана Борисівна');

--

-- Тригери `course`

--

DELIMITER $$

CREATE TRIGGER `tr\_insert\_professorname` BEFORE UPDATE ON `course` FOR EACH ROW BEGIN

SET NEW.prof\_username = (SELECT prof\_username FROM professor WHERE professor.professor\_id = NEW.professor\_id);

END

$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблиці `exam`

--

CREATE TABLE `exam` (

`exam\_id` int(11) NOT NULL,

`student\_id` int(11) DEFAULT NULL,

`course\_id` smallint(6) DEFAULT NULL,

`date` date DEFAULT NULL,

`grade` decimal(10,0) DEFAULT NULL,

`prof\_username` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`coursename` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Дамп даних таблиці `exam`

--

INSERT INTO `exam` (`exam\_id`, `student\_id`, `course\_id`, `date`, `grade`, `prof\_username`, `coursename`) VALUES

(10001, 13, 653, '2023-12-21', '65', 'Лучшева Оксана Вадимівна', 'Програмування мовою Python'),

(10002, 35, 657, '2023-12-26', '65', 'Клименко Тетяна Анатоліївна', 'Оптимізація та просування сайтів (SEO)'),

(10003, 13, 322, '2023-12-18', '95', 'Губка Сергій Олексійович', 'Операційні системи'),

(10004, 13, 653, '2023-12-26', '95', 'Лучшева Оксана Вадимівна', 'Програмування мовою Python'),

(10005, 5, 671, '2023-12-27', '86', 'Клименко Тетяна Анатоліївна', 'Системи реального часу'),

(10006, 13, 659, '2023-12-20', '100', 'Вдовітченко Олександр Валерійович', 'Web-програмування мовою PHP');

--

-- Тригери `exam`

--

DELIMITER $$

CREATE TRIGGER `trg\_update\_course\_info\_in\_exam` BEFORE INSERT ON `exam` FOR EACH ROW BEGIN

SET NEW.coursename = (SELECT coursename FROM course WHERE course.course\_id = NEW.course\_id);

SET NEW.prof\_username = (SELECT prof\_username FROM course WHERE course.course\_id = NEW.course\_id);

END

$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблиці `faculty`

--

CREATE TABLE `faculty` (

`faculty\_id` tinyint(4) NOT NULL,

`facultyname` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Дамп даних таблиці `faculty`

--

INSERT INTO `faculty` (`faculty\_id`, `facultyname`) VALUES

(1, 'Факультет літакобудування'),

(2, 'Факультет авіаційних двигунів'),

(3, 'Факультет систем управління літальних апаратів'),

(4, 'Факультет ракетно-космічної техніки'),

(5, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(6, 'Факультет програмної інженерії та бізнесу'),

(7, 'Гуманітарно-правовий факультет');

-- --------------------------------------------------------

--

-- Структура таблиці `group`

--

CREATE TABLE `group` (

`group\_id` smallint(6) NOT NULL,

`group\_number` varchar(100) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`faculty\_id` tinyint(4) DEFAULT NULL,

`facultyname` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Дамп даних таблиці `group`

--

INSERT INTO `group` (`group\_id`, `group\_number`, `faculty\_id`, `facultyname`) VALUES

(430, '430', 4, 'Факультет ракетно-космічної техніки'),

(522, '522-ст', 5, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(526, '526-ст', 5, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(539, '539-а', 5, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(611, '611п', 6, 'Факультет програмної інженерії та бізнесу'),

(631, '631п', 6, 'Факультет програмної інженерії та бізнесу'),

(632, '632п', 6, 'Факультет програмної інженерії та бізнесу'),

(633, '633п', 6, 'Факультет програмної інженерії та бізнесу'),

(634, '634п', 6, 'Факультет програмної інженерії та бізнесу');

--

-- Тригери `group`

--

DELIMITER $$

CREATE TRIGGER `tr\_insert\_group\_facultyname` BEFORE INSERT ON `group` FOR EACH ROW BEGIN

SET NEW.facultyname = (SELECT facultyname FROM faculty WHERE faculty.faculty\_id = NEW.faculty\_id);

END

$$

DELIMITER ;

DELIMITER $$

CREATE TRIGGER `trg\_prevent\_delete\_group\_with\_students` BEFORE DELETE ON `group` FOR EACH ROW BEGIN

IF EXISTS (SELECT 1 FROM student WHERE group\_id = OLD.group\_id) THEN

SIGNAL SQLSTATE '45000'

SET MESSAGE\_TEXT = 'У групі є студенти.';

END IF;

END

$$

DELIMITER ;

-- --------------------------------------------------------

--

-- Структура таблиці `professor`

--

CREATE TABLE `professor` (

`professor\_id` int(11) NOT NULL,

`prof\_username` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Дамп даних таблиці `professor`

--

INSERT INTO `professor` (`professor\_id`, `prof\_username`) VALUES

(972, 'Качуров Євгеній Васильович'),

(973, 'Жарікова Світлана Борисівна'),

(974, 'Дегтярьова Тетяна Григорівна'),

(975, 'Вдовітченко Олександр Валерійович'),

(976, 'Клименко Тетяна Анатоліївна'),

(977, 'Лучшева Оксана Вадимівна'),

(978, 'Попов Анатолій Владиславович'),

(979, 'Клименко Ігор Андрійович'),

(980, 'Шпілінська Ольга Леонідівна'),

(981, 'Івановська Ольга Володимирівна'),

(982, 'Губка Сергій Олексійович'),

(983, 'Заболотний Олександр Віталійович'),

(984, 'Петухов Ілля Іванович'),

(985, 'Зеленський Роман Леонідович'),

(986, 'Горлов Олександр Кузьмич'),

(987, 'Брега Дмитро Андрійович');

-- --------------------------------------------------------

--

-- Структура таблиці `student`

--

CREATE TABLE `student` (

`student\_id` int(11) NOT NULL,

`username` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL,

`grades\_record\_number` int(11) DEFAULT NULL,

`group\_id` smallint(6) DEFAULT NULL,

`facultyname` varchar(255) COLLATE utf8mb4\_unicode\_ci DEFAULT NULL

) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4\_unicode\_ci;

--

-- Дамп даних таблиці `student`

--

INSERT INTO `student` (`student\_id`, `username`, `grades\_record\_number`, `group\_id`, `facultyname`) VALUES

(0, 'Петренко Євген Євгенович', 12345, 631, 'Факультет програмної інженерії та бізнесу'),

(1, 'Бондарев Віктор Вячеславович', 1411238, 631, 'Факультет програмної інженерії та бізнесу'),

(2, 'Вестніков Владислав Михайлович', 9622950, 632, 'Факультет програмної інженерії та бізнесу'),

(3, 'Гальцов Ілля Миколайович', 9622971, 633, 'Факультет програмної інженерії та бізнесу'),

(4, 'Деліжан Ірина Сергіївна', 9639286, 634, 'Факультет програмної інженерії та бізнесу'),

(5, 'Демиденко Катерина Віталіївна', 9622975, 522, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(6, 'Єфременко Богдан Вячеславович', 9639295, 526, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(7, 'Замоздра Владислав Олександрович', 9622956, 539, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(8, 'Коляндра Юлія Миколаївна', 9639279, 430, 'Факультет ракетно-космічної техніки'),

(9, 'Кузьмичов Ігор Ігорович', 9639308, 631, 'Факультет програмної інженерії та бізнесу'),

(10, 'Лагута Владислав Дмитрович', 9622967, 632, 'Факультет програмної інженерії та бізнесу'),

(11, 'Макарук Анастасія Миколаївна', 9622942, 633, 'Факультет програмної інженерії та бізнесу'),

(12, 'Марченко Владислав Віталійович', 9639271, 634, 'Факультет програмної інженерії та бізнесу'),

(13, 'Міхов Андрій Сергійович', 9639290, 522, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(14, 'Московець Федір Андрійович', 9622972, 526, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(15, 'Оніщенко Оксана Олександрівна', 9639277, 539, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(16, 'Полежай Павло Дмитрович', 9639272, 430, 'Факультет ракетно-космічної техніки'),

(17, 'Ракін Олексій Дмитрович', 9622944, 631, 'Факультет програмної інженерії та бізнесу'),

(18, 'Ремесло Олександр Олександрович', 10508273, 632, 'Факультет програмної інженерії та бізнесу'),

(19, 'Рєпіна Аліна Євгенівна', 9622981, 633, 'Факультет програмної інженерії та бізнесу'),

(20, 'Саранча Євгеній Ігорович', 9622943, 634, 'Факультет програмної інженерії та бізнесу'),

(21, 'Севастюк Сергій Сергійович', 9622969, 522, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(22, 'Ткаченко Єгор Русланович', 9622945, 526, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(23, 'Цикаленко Дмитро Геннадійович', 9878602, 539, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(24, 'Чернищук Сергій Володимирович', 8080680, 430, 'Факультет ракетно-космічної техніки'),

(25, 'Антипенко Антон Євгенович', 9639280, 631, 'Факультет програмної інженерії та бізнесу'),

(26, 'Антоненко Валерія Артемівна', 9639302, 632, 'Факультет програмної інженерії та бізнесу'),

(27, 'Бережний Віталій Віталійович', 9639284, 633, 'Факультет програмної інженерії та бізнесу'),

(28, 'Бондар Нікіта Олексійович', 9639285, 634, 'Факультет програмної інженерії та бізнесу'),

(29, 'Вечтомова Дарина Володимирівна', 9639304, 522, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(30, 'Вітковський Сергій Володимирович', 8163818, 526, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(31, 'Головкіна Тетяна Андріївна', 9639282, 539, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(32, 'Голубков Роман Васильович', 9622987, 430, 'Факультет ракетно-космічної техніки'),

(33, 'Гончарук Наум Сергійович', 9639270, 631, 'Факультет програмної інженерії та бізнесу'),

(34, 'Гула Єгор Спартакович', 9622973, 632, 'Факультет програмної інженерії та бізнесу'),

(35, 'Мацкевич Демид Данилович', 9622958, 633, 'Факультет програмної інженерії та бізнесу'),

(36, 'Москвітін Юрій Костянтинович', 9623004, 634, 'Факультет програмної інженерії та бізнесу'),

(37, 'Мушка Яків Олегович', 9622949, 522, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(38, 'Ортинський Богдан Олександрович', 9622991, 526, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(39, 'Пеньшин Єгор Миколайович', 9639291, 539, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(40, 'Середа Єгор Ігорович', 9622957, 430, 'Факультет ракетно-космічної техніки'),

(41, 'Сех Олександр Володимирович', 9622983, 631, 'Факультет програмної інженерії та бізнесу'),

(42, 'Славков Денис Анатолійович', 9622948, 631, 'Факультет програмної інженерії та бізнесу'),

(43, 'Фоменко Катерина Володимирівна', 9639278, 633, 'Факультет програмної інженерії та бізнесу'),

(44, 'Царевський Даніїл Сергійович', 9639289, 634, 'Факультет програмної інженерії та бізнесу'),

(45, 'Шевко Валентин Сергійович', 106127, 539, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(46, 'Шепель Олександр Володимирович', 9623003, 526, 'Факультет радіоелектроніки, комп\'ютерних систем та інфокомунікацій'),

(47, 'Бондарев Віктор Вячеславович', 1411238, 632, 'Факультет програмної інженерії та бізнесу');

-- --------------------------------------------------------

--

-- Дублююча структура для представлення `track\_view`

-- (Див. нижче для фактичного подання)

--

CREATE TABLE `track\_view` (

`username` varchar(255)

,`grades\_record\_number` int(11)

,`group\_id` smallint(6)

,`exam\_id` int(11)

,`coursename` varchar(255)

,`date` date

,`grade` decimal(10,0)

,`prof\_username` varchar(255)

);

-- --------------------------------------------------------

--

-- Структура для представлення `track\_view`

--

DROP TABLE IF EXISTS `track\_view`;

CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIEW `track\_view` AS select `student`.`username` AS `username`,`student`.`grades\_record\_number` AS `grades\_record\_number`,`student`.`group\_id` AS `group\_id`,`exam`.`exam\_id` AS `exam\_id`,`exam`.`coursename` AS `coursename`,`exam`.`date` AS `date`,`exam`.`grade` AS `grade`,`exam`.`prof\_username` AS `prof\_username` from (`student` join `exam` on((`student`.`student\_id` = `exam`.`student\_id`))) ;

--

-- Індекси збережених таблиць

--

--

-- Індекси таблиці `course`

--

ALTER TABLE `course`

ADD PRIMARY KEY (`course\_id`),

ADD KEY `fk\_course\_proffesor` (`professor\_id`);

--

-- Індекси таблиці `exam`

--

ALTER TABLE `exam`

ADD PRIMARY KEY (`exam\_id`),

ADD KEY `fk\_exam\_course` (`course\_id`),

ADD KEY `fk\_exam\_student` (`student\_id`);

--

-- Індекси таблиці `faculty`

--

ALTER TABLE `faculty`

ADD PRIMARY KEY (`faculty\_id`);

--

-- Індекси таблиці `group`

--

ALTER TABLE `group`

ADD PRIMARY KEY (`group\_id`),

ADD KEY `fk\_group\_faculty` (`faculty\_id`);

--

-- Індекси таблиці `professor`

--

ALTER TABLE `professor`

ADD PRIMARY KEY (`professor\_id`);

--

-- Індекси таблиці `student`

--

ALTER TABLE `student`

ADD PRIMARY KEY (`student\_id`),

ADD KEY `fk\_student\_group` (`group\_id`);

--

-- Обмеження зовнішнього ключа збережених таблиць

--

--

-- Обмеження зовнішнього ключа таблиці `course`

--

ALTER TABLE `course`

ADD CONSTRAINT `fk\_course\_proffesor` FOREIGN KEY (`professor\_id`) REFERENCES `professor` (`professor\_id`);

--

-- Обмеження зовнішнього ключа таблиці `exam`

--

ALTER TABLE `exam`

ADD CONSTRAINT `fk\_exam\_course` FOREIGN KEY (`course\_id`) REFERENCES `course` (`course\_id`),

ADD CONSTRAINT `fk\_exam\_student` FOREIGN KEY (`student\_id`) REFERENCES `student` (`student\_id`);

--

-- Обмеження зовнішнього ключа таблиці `group`

--

ALTER TABLE `group`

ADD CONSTRAINT `fk\_group\_faculty` FOREIGN KEY (`faculty\_id`) REFERENCES `faculty` (`faculty\_id`);

--

-- Обмеження зовнішнього ключа таблиці `student`

--

ALTER TABLE `student`

ADD CONSTRAINT `fk\_student\_group` FOREIGN KEY (`group\_id`) REFERENCES `group` (`group\_id`);

COMMIT;

/\*!40101 SET CHARACTER\_SET\_CLIENT=@OLD\_CHARACTER\_SET\_CLIENT \*/;

/\*!40101 SET CHARACTER\_SET\_RESULTS=@OLD\_CHARACTER\_SET\_RESULTS \*/;

/\*!40101 SET COLLATION\_CONNECTION=@OLD\_COLLATION\_CONNECTION \*/;