МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ФРАНКА
ФАКУЛЬТЕТ УПРАВЛІННЯ ФІНАНСАМИ ТА БІЗНЕСУ
Кафедра цифрової економіки та бізнес аналізу

Курсова робота

З дисципліни «Проектування та адміністрування БД і СД»

На тему:

«Інформаційна система «Деканат»»

спеціальність: економіка

спеціалізація: інформаційні технології в бізнесі

освітній ступінь: бакалавр

Науковий керівник:
к.ф.-м.н., доц. Депутат Б.Я
(прізвище, ім'я, по-батькові)
_____ (підпис)

“___” _____ 2021 р.

Виконавець:
Березюк Назар Любомирович
(прізвище, ім'я, по-батькові)
УФЕ-31с група
_____ (підпис)

“___” _____ 2021 р.

Загальна кількість балів _____
(підписи, піп членів комісії)

Львів 2021

ЗМІСТ

ВСТУП

Актуальність теми. Інформаційна система «Деканат» - організаційний центр, котрий з’єднює студентів та викладачів. Студенти у системі «Деканат» можуть переглянути відомості про навчання, знайти свій середній бал, дізнатися заборгованості та бути проінформованим щодо студентського життя в університеті.

Актуальність даної теми стає популярнішою на початку 2020 року, коли світ покрила пандемія COVID-19. Саме у цей період університетське життя переходить на дистанційну форму навчання. «Деканат» стає потрібним, щоб дізнатися бали про поточну успішність, знайти дати залікових іспитів та екзаменаційні дати, можна подивитися з яких предметів потрібно відпрацювати пропуски та які лабораторні чи семінарські завдання потрібно доздати.

Інформаційна система «Деканат» дозволяє студенту ознайомитися із поточною успішністю, семестровими балами, статистикою оцінок, пропусками занять, практикою, вибірковими дисциплінами, науковою діяльністю студента, подяками чи доганами, а також індивідуальним навчальним планом. Викладач чи студент може ознайомитися з останніми подіями через онлайн дошку оголошень. Зручним доповненням є те, що викладачі і студенти мають змогу переглянути свій розклад занять та отримувати інформаційні повідомлення у системі «Деканат».

Через пандемію COVID-19, дистанційне навчання та запроваджений карантин, інформаційна система «Деканат» стає незамінним веб ресурсом для студентів та викладачів, допомагаючи організувати онлайн навчання.

Мета і завдання дослідження. Метою даної роботи є обґрунтування теоретичних основ та реалізація бази даних для інформаційної системи «Деканат». В даній роботі наведені короткі відомості про основні засоби, які потрібні для реалізації системи «Деканат» і опис структури бази даних. На початковому етапі практичної частини розроблялась база даних на платформі

Microsoft SQL. Наступним і завершальним етапом була побудова бази даних, її наповнення, розроблення зв’язків між таблицями та створення запитів.

Для вирішення даної мети були поставленні наступні завдання:

1. Проаналізувати предметну область інформаційної системи «Деканат»
2. Описати всі аспекти використання баз даних
3. Відобразити інформаційну систему за допомогою запитів

Об’єктом дослідження виступає база даних, як основний елемент інформаційної системи книгарні

Основним предметом дослідження є програмне середовище, яке застосовується для автоматизації онлайн навчання для студентів і викладачів.

Використане програмне забезпечення. Для досягнення результату використовувалось програмне середовище Microsoft SQL.

Структура роботи. Курсова складається з двох розділів («Аналіз предметної області», «Розробка бази даних»), а також з висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття та визначення

База даних (БД) — впорядкований набір логічно взаємопов'язаних даних, що використовується спільно, та призначений для задоволення інформаційних потреб користувачів. У технічному розумінні включно й система керування БД.

Головним завданням БД є гарантоване збереження значних обсягів інформації та надання доступу до неї користувачеві або ж прикладній програмі. Таким чином БД складається з двох частин : збереженої інформації та системи управління нею. З метою забезпечення ефективності доступу записи даних організовують як множину фактів (елемент даних).

Дані – це інформація, відомості, показники, необхідні для ознайомлення з ким-, чим-небудь, для характеристики когось, чогось або для прийняття певних висновків, рішень. В базах даних важливу роль відіграє інформація - Інформація — абстрактне поняття, що має різні значення залежно від контексту.

Кожна створена база даних має свою предметну область. Предметна область – це необхідний для розробки бази даних об’єкт, який має в собі дані, які будуть зберігатися в базі даних.

Модель даних — абстрактне представлення реального світу, що відображає тільки ті об'єкти, що безпосередньо стосуються програми. Це, як правило, визначає специфічну групу об'єктів, їх атрибутивне значення і відношення між ними.

Відомі два підходи до організації інформаційних масивів: файлова організація та організація у вигляді бази даних. Файлова організація передбачає спеціалізацію та збереження інформації, орієнтованої, як правило, на одну прикладну задачу, та забезпечується прикладним програмістом. Така організація дозволяє досягнути високої швидкості обробки інформації, але

характеризується рядом недоліків. Оскільки керування здійснюється різними особами (групами осіб), відсутня можливість виявити порушення суперечливості збереженої інформації. Розроблені файли для спеціалізованих прикладних програм не можна використовувати для задоволення запитів користувачів, які перекривають дві і більше області. Крім того, файлова організація даних внаслідок відмінностей структури записів і форматів передання даних не забезпечує виконання багатьох інформаційних запитів навіть у тих випадках, коли всі необхідні елементи даних містяться в наявних файлах. Тому виникає необхідність відокремити дані від їхнього опису, визначити таку організацію збереження даних з обліком існуючих зв'язків між ними, яка б дозволила використовувати ці дані одночасно для багатьох застосувань. Вказані причини обумовили появу баз даних. База даних може бути визначена як структурна сукупність даних, що підтримуються в активному стані та відображає властивості об'єктів зовнішнього (реального) світу. В базі даних містяться не тільки дані, але й описи даних, і тому інформація про форму зберігання вже не схована в сполученні "файл-програма", вона явним чином декларується в базі.

База даних орієнтована на інтегровані запити, а не на одну програму, яку випадку файлового підходу, і використовується для інформаційних потреб багатьох користувачів. В зв'язку з цим бази даних дозволяють в значній мірі скоротити надлишковість інформації. Перехід від структури БД до потрібної структури в програмі користувача відбувається автоматично за допомогою систем управління базами даних (СУБД).

Системи управління базами даних – це програмні засоби, за допомогою яких можна створювати бази даних, заповнювати їх та працювати з ними. У світі існує багато різноманітних систем управління базами даних. Багато з них насправді є не закінченими продуктами, а спеціалізованими мовами програмування, за допомогою яких кожний, хто вивчить дану мову, може сам створювати такі структури, які йому потрібні, і вводити в них необхідні

елементи управління. До таких мов відносяться Clipper, Paradox, FoxPro та інші.

1.2 Призначення та класифікація систем управління базами даних

Система управління базами даних (СУБД) — комп'ютерна програма чи комплекс програм, що забезпечує користувачам можливість створення, збереження, оновлення, пошук інформації та контролю доступу в базах даних.

СУБД - це складна програмна система накопичення та з наступним маніпулюванням даними, що представляють інтерес для користувача. Кожній прикладній програмі СУБД надає інтерфейс з базою даних та має засоби безпосереднього доступу до неї. Таким чином, СУБД відіграє центральну роль в функціонуванні автоматизованого банку даних.

Архітектурно СУБД складається з двох великих компонент. За допомогою мови опису даних (МОД) створюються описи елементів, груп та записів даних, а також взаємозв'язки між ними, які, як правило, задаються у вигляді таблиць. В залежності від конкретної реалізації СУБД мову опису даних підрозділяють на мову опису схеми бази даних (МОС) та мову опису підсхем бази даних (МОП).

Для виконання операцій з базою даних в прикладних програмах використовується мова маніпулювання даними (ММД). Фактична структура фізичного зберігання даних відома тільки СУБД.

З метою забезпечення зв'язків між програмами користувачів і СУБД (що особливо важливо при мультипрограмному режимі роботи операційної системи) в СУБД виділяють особливу складову - резидентний модуль системи керування базами даних. Цей модуль значно менший від всієї СУБД , тому на час функціонування автоматизованого банку інформації він може постійно знаходитись в основній пам'яті ЕОМ та забезпечувати взаємодію всіх складових СУБД і програм, які до неї звертаються.

РОЗДІЛ 2. РОЗРОБКА БАЗИ ДАНИХ.

2.1 Основні відомості про бази даних

З розвитком інформаційного забезпечення систем автоматизованої обробки інформації, прагненням забезпечити виконання нових режимів обробки даних у реальному часі і з мультидоступом до схованих даних позначилась нова тенденція до складення інформаційного забезпечення розподілених баз даних. В умовах використання таких баз створюються комплексні масиви нелінійної структури, які мають усі дані про ту чи іншу предметну область або про керований об'єкт як постійного, так і перемінного характеру.

Взагалі база даних є сукупність даних на машинних носіях, які використовуються при функціонуванні системи обробки інформації, організовані по визначеним правилам, які передбачають загальні принципи описування збереження і маніпулювання ними, а також які незалежні від прикладних програм. В основі організації бази даних є модель даних, яка визначає правила, у відповідності з якими структуруються дані. За допомогою моделі представляється велика кількість даних і описуються взаємно зв’язки між ними. Найбільш поширені такі моделі даних: ієрархічна, сітьова, реляційна.

В ієрархічній моделі зв'язок даних "один до одного" (1:1) означає, що кожному значенню (екземпляру) елемента даних А відповідає одне і тільки одне значення, пов'язаного з ним елемента В. Наприклад, поміж такими елементами пар даних, як код готової продукції і її найменуванням є вищезазначений зв'язок, так як тільки кожному коду продукції відповідає одне її найменування.

Зазначимо, що ієрархічна модель даних будується на основі принципу підпорядкованості поміж елементами даних і представляє собою деревоподібну структуру, яка складається із вузлів (сегментів) і дуг (гілок).

Дерево у ієрархічній структурі упорядковане за існуючими правилами розташування його сегментів і гілок: на верхньому рівні знаходиться один, кореневий (вихідний) сегмент, сегмент другого рівня, породжений, залежить від першого, вихідного; доступ до кожного породженого (крім кореневого) здійснюється через його вихідний сегмент; кожний сегмент може мати по декілька екземплярів конкретних значень елементів даних, а кожний елемент породженого сегменту пов’язаний з екземпляром вихідного і створює один логічний запис; екземпляр породженого сегменту не може існувати самостійно, тобто без кореневого сегменту; при вилученні екземпляру кореневого сегмента також вилучаються усі підпорядковані і взаємопов'язані з ним екземпляри породжених сегментів.

В сітьовій моделі зв'язок "один до багатьох" (1:В) означає, що значенню елемента А відповідають багато (більше одного) значень, пов'язанню з ним елементів В. Наприклад, поміж елементами даних "код виробу" (елемент А) і "кодом матеріалів" (елементи В) існує такий взаємозв'язок бо на виготовлення одного виробу використовується багато різних матеріалів.

Сітьова модель даних представляє собою орієнтований граф з пойменованими вершинами і дугами. Вершини графа - записи, які представляють собою по іменовану сукупність логічних взаємозв'язаних елементів даних або агрегатів даних. Під агрегатом даних розуміють пошановану сукупність елементів даних, які є усередині запису. Для кожного типу записів може бути кілька екземплярів конкретних значень його інформаційних елементів Два записи, взаємозв'язані дугою, створюють набір даних. Запис, з якого виходить дуга, називається власником набору, а запис, до якого вона направлена, - членом набору.

В реляційній моделі зв'язок "багатьох до багатьох" (В:В) указує на те, що декільком значенням елементів даних А відповідає декілька значені елементів даних В. Наприклад, поміж елементами даних "код операції технологічного процесу" і "табельний номер працівника" існує зазначені взаємозв'язок, так як багато операцій технологічного процесу можуть виконувати різні працівники

(табельні номери) і навпаки.

Реляційна модель даних являє собою набір двомірних плоских таблиць, що складаються з рядків і стовпців. Первинний документ або лінійний массив являє собою плоску двомірну таблицю. Така таблиця називається відношенням, кожний стовбець-атрибутом, сукупність значень одного типу (стовпця) –доменом, а рядка – кортежем. Таким чином, стовпці таблиці являються традиційними елементами даних, а рядки – записами. Таблиці (відношення) мають імена. Імена також присвоюються і стовпцям таблиці. Кожний кортеж (запис) відношення має ключ. Ключі є прості і складні. Простий ключ-це ключ, який складається з одного атомарного атрибуту, значення якого унікальне (яке не повторюються).Складний ключ складається з двох і більше атрибутів. Для зв'язків відношень друг з другом в базі даних є зовнішні ключі. Атрибут або комбінація атрибута відношення є зовнішнім ключем, якщо він не є основним (первинним) ключем цього відношення, але являється первинним ключем для другого відношення.

Первинний ключ — атрибут, або набір атрибутів, що однозначно ідентифікує кортеж даного відношення. Первинний ключ обов'язково унікальний, він єдиний і найголовніший із унікальних ключів.

В реляційних базах даних первинний ключ задається обмеженням PRIMARY KEY.

Зовнішній (вторинний) ключ — це одне або кілька полів (стовпців) у таблиці, що містять посилання на поле або поля первинного ключа в іншій таблиці. Зовнішній ключ визначає спосіб об'єднання таблиць.

З двох логічно пов'язаних таблиць одну називають таблицею первинного ключа або головною таблицею, а іншу таблицею вторинного (зовнішнього) ключа або підпорядкованою таблицею. СУБД дозволяють зіставити споріднені записи з обох таблиць і спільно вивести їх у формі, звіті або запиті.

Існує три типи первинних ключів: ключові поля лічильника (лічильник), простий ключ і складовий ключ.

Поле лічильника (Тип даних «Счетчик»). Для кожного запису цього поля таблиці автоматично заноситься унікальне числове значення.

Простий ключ. Якщо поле містить унікальні значення, такі як коди чи інвентарні номери, то це поле можна визначити як первинний ключ. В якості ключа можна визначити всі поля, що містить дані, якщо це поле не містить повторювані значення або значення Null.

Складові ключі. У випадках, коли неможливо гарантувати унікальність значень кожного поля, існує можливість створити ключ, що складається з декількох полів. Найчастіше така ситуація виникає для таблиці, використовуваної для зв'язування двох таблиць відношенням «багато — до — багатьох».

Необхідно ще раз відзначити, що в полі первинного ключа повинні бути тільки унікальні значення в кожному рядку таблиці, тобто збіг не допускається, а в полі вторинного або зовнішнього ключа збіг значень у рядках таблиці допускається.

Якщо виникають труднощі з вибором потрібного типу первинного ключа, то в якості ключа доцільно вибрати поле лічильника.

Різновидністю баз даних, з точки зору їх зберігання і використання, є розподіленні бази даних. Ці бази даних широко використовуються при організації комплексів взаємопов'язаних АРМ фахівців, на яких застосовуються ПЕОМ .

Розподілена база даних - це сукупність логічно зв'язаних баз даних або частин однієї бази, які розділені поміж декількома територіально – розподіленими ПЕОМ і забезпечені відповідними можливостями для управління цими базами або їх частинами. Тобто, розподілена база даних реалізується на різних просторово розосереджених обчислювальних засобах, разом з організаційними, технічними і програмними засобами її створення і ведення.

2.2 Нормалізація бази даних

Нормалізація схеми бази даних — покроковий процес розбиття одного відношення (на практиці: таблиці) відповідно до алгоритму нормалізації на декілька відношень на базі функціональних залежностей.

Нормальна форма — властивість відношення в реляційній моделі даних, що характеризує його з точки зору надмірності, яка потенційно може призвести до логічно помилкових результатів вибірки або зміни даних. Нормальна форма визначається як сукупність вимог, яким має задовольняти відношення.

Таким чином, схема реляційної бази даних переходить у першу, другу, третю і так далі нормальні форми. Якщо відношення відповідає критеріям нормальної форми n та всіх попередніх нормальних форм, тоді вважається, що це відношення знаходиться у нормальній формі рівня n.

Перша нормальна форма (1НФ, 1NF) утворює ґрунт для структурованої схеми бази даних:

Кожна таблиця повинна мати основний ключ: мінімальний набір колонок, які ідентифікують запис.

Уникнення повторень груп (категорії даних, що можуть зустрічатись різну кількість разів в різних записах) правильно визначаючи неключові атрибути.

Атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень.

Друга нормальна форма

Друга нормальна форма (2НФ, 2NF) вимагає, аби дані, що зберігаються в таблицях із композитним ключем, не залежали лише від частини ключа:

Схема бази даних повинна відповідати вимогам першої нормальної форми.

Дані, що повторно з'являються в декількох рядках, виносяться в окремі таблиці.

Третя нормальна форма

Третя нормальна форма (3НФ, 3NF) вимагає, аби дані в таблиці залежали винятково від основного ключа:

Схема бази даних повинна відповідати всім вимогам другої нормальної форми.

Будь-яке поле, що залежить від основного ключа та від будь-якого іншого поля, має виноситись в окрему таблицю.

Нормальна форма Бойса — Кодда

Відношення знаходиться в НФБК тоді і лише тоді, коли детермінант кожної функціональної залежності є потенційним ключем. Якщо це правило не виконується, то, щоб привести вказане відношення до НФБК, його слід розділити на два відношення шляхом двох операцій проекції на кожну функціональну залежність, детермінант якої не є потенційним ключем:

Проекція без атрибутів залежної частини такої функціональної залежності;

Проекція на всі атрибути цієї функціональної залежності.

Визначення НФБК не потребує жодних умов попередніх нормальних форм. Якщо проводити нормалізацію послідовно, то в переважній більшості випадків при досягненні 3НФ автоматично будуть задовольнятися вимоги НФБК. 3НФ не збігається з НФБК лише тоді, коли одночасно виконуються такі 3 умови:

- Відношення має 2 або більше потенційних ключів.
- Ці потенційні ключі складені (містять більш ніж один атрибут)
- Ці потенційні ключі перекриваються, тобто мають щонайменше один спільний атрибут.

Четверта нормальна форма

Четверта нормальна форма (4НФ, 4NF) потребує, аби в схемі баз даних не було нетривіальних багатозначних залежностей множин атрибутів від будь чого, окрім надмножини ключа-кандидата. Вважається, що таблиця знаходиться у 4НФ тоді і лише тоді, коли вона знаходиться в НФБК та

багатозначні залежності є функціональними залежностями. Четверта нормальна форма усуває небажані структури даних — багатозначні залежності.

П'ята нормальна форма

П'ята нормальна форма (5НФ, 5NF, PJ/NF) вимагає, аби не було нетривіальних залежностей об'єднання, котрі б не витікали із обмежень ключів. Вважається, що таблиця в п'ятій нормальній формі тоді і лише тоді, коли вона знаходиться в 4НФ та кожна залежність об'єднання зумовлена її ключами-кандидатами.

Нормальна форма домен/ключ

Ця нормальна форма вимагає, аби в схемі не було інших обмежень окрім ключів та доменів.

Шоста нормальна форма

Таблиця знаходиться у 6NF, якщо вона знаходиться у 5NF та задовольняє вимозі відсутності нетривіальних залежностей. Зазвичай 6NF ототожнюють з DKNF.

2.3 Проектування структури бази даних

Для розробки моделі бази даних обрана реляційна модель даних.

Переваги реляційної моделі даних:

- простота і доступність для розуміння користувачем. Єдиною використовуваною інформаційною конструкцією є «таблиця»;
- суворі правила проектування, які базуються на математичному апараті;
- повна незалежність даних. Зміни в прикладній програмі при зміні реляційної БД мінімальні;
- для організації запитів і написання прикладного ПЗ немає необхідності знати конкретну організацію БД у зовнішній пам'яті.

Недоліки реляційної моделі:

- далеко не завжди предметна область може бути представлена у вигляді «таблиць»;
- в результаті логічного проектування з'являється множина «таблиць». Це призводить до труднощів розуміння структури даних;
- БД займає відносно багато зовнішньої пам'яті;
- відносно низька швидкість доступу до даних.

В процесі проектування структури бази даних потрібно створити діаграму концептуальної моделі даних. На основі визначених елементів і зв’язків створити ER – діаграму.

Діаграма концептуальної моделі даних представлена на рисунку 2.1

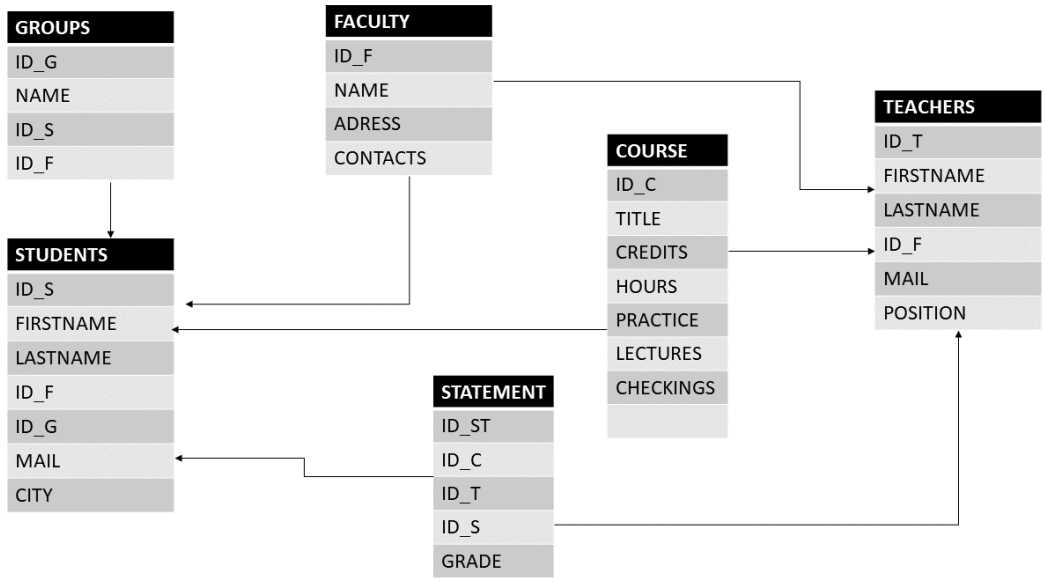


Рис.2.1 Зображення ER-діаграми

На основі даної ER-діаграми будується реляційна модель даних.

Поля які ідентифікують стовпці таблиці STUDENTS:

- ID_S - ідентифікаційний код студента;
- FIRSTNAME – ім’я студента;
- LASTNAME – прізвище студента;
- ID_F – ідентифікаційний код факультету, де навчається студент;
- ID_G – ідентифікаційний код групи;
- MAIL – поштова скринька студента;

- CITY – місто, де проживає студент;
- POSITION – роль, яку відіграє студент у групі;

Поля які ідентифікують таблицю STATEMENT:

- ID_ST – ідентифікаційний код відомості;
- ID_C - ідентифікаційний код навчальної дисципліни;
- ID_T – ідентифікаційний код викладача;
- ID_S – ідентифікаційний код студента;
- GRADE - оцінка;

Поля які ідентифікують таблицю COURSE:

- ID_C - ідентифікаційний код навчальної дисципліни;
- TITLE – назва навчальної дисципліни;
- CREDITS – кількість кредитів;
- HOURS – кількість годин;
- PRACTICE – кількість годин практики;
- LECTURES – кількість годин лекцій;
- CHECKINGS – перевірка якості знань;

Поля які ідентифікують таблицю TEACHERS:

- ID_T – ідентифікаційний код викладача;
- FIRSTNAME – ім’я викладача;
- LASTNAME – прізвище викладача;
- ID_F – ідентифікаційний код факультету;
- MAIL – поштова скринька викладача;
- POSITION – роль викладача;

Поля які ідентифікують таблицю FACULTY:

- ID_F – ідентифікаційний код факультету;
- NAME – назва факультету;
- ADRESS – адреса факультету;
- CONTACTS – контактні дані факультету;

Поля які ідентифікують таблицю GROUPS:

- ID_G – ідентифікаційний код групи;

- NAME – назва групи;
- ID_S – ідентифікаційний код студента;
- ID_F – ідентифікаційний код факультету;

2.4 Визначення типів даних

SQL Server - система управління реляційними базами даних Microsoft . Це повнофункціональна база даних, в основному створена для конкуренції конкурентам Oracle Database (DB) та MySQL.

MSSQL - це звичайний каталог, що містить файли певного формату - таблиці. Таблиці складаються із записів, а записи, у свою чергу, складаються з полів. Поле має два атрибути - ім'я і тип даних.

Найбільш використовувані типи даних:

- VARCHAR - може зберігати не більше 255 символів. На відміну від CHAR - для зберігання значення даного типу виділяється необхідна кількість пам'яті;
- TEXT - може зберігати не більше 65 535 символів;
- INT - діапазон від -2 147 483 648 до 2 147 483 647;
- DATE - зберігає дати від 0001-01-01 (1 січня 0001 року) до 9999-12-31 (31 грудня 9999 року);
- TIME - зберігає час в діапазоні від 00: 00: 00.0000000 до 23: 59: 59.9999999;
- XML - зберігає документи XML або фрагменти документів XML.

2.5 Реалізація SQL-скрипту

Для того щоб створити базу даних DECANAT потрібно прописати у програмному середовищі Microsoft SQL наступні слова:

CREATE DATABASE DECANAT

Якщо потрібно створити таблицю STUDENTS, котра міститиме в собі стовпці (ID_S, FIRSTNAME, LASTNAME, ID_F, ID_G, MAIL, ADRESS , POSITION) слід написати:

CREATE TABLE STUDENTS(ID_S INT, FIRSTNAME VARCHAR(30), LASTNAME VARCHAR(30), ID_F INT, ID_G INT, MAIL VARCHAR(30), CITY VARCHAR(40), POSITION VARCHAR(30), PRIMARY KEY(ID_S))

Також у даному рядку вказано, що PRIMARY KEY (первинний ключ) для даної таблиці буде стовпець ID_S.

Аналогічно створюються таблиці Teachers, Groups, Faculty, Course, Statement.

CREATE TABLE TEACHERS (ID_T INT, FIRSTNAME VARCHAR(30), LASTNAME VARCHAR(30), ID_F INT, MAIL VARCHAR(30),POSITION VARCHAR(30), PRIMARY KEY(ID_T))

CREATE TABLE GROUPS(ID_G INT, NAME VARCHAR(30), ID_S INT, ID_F INT, PRIMARY KEY(ID_G))

CREATE TABLE FACULTY(ID_F INT, NAME VARCHAR(30), ADRESS VARCHAR(50), CONTACTS VARCHAR(100), PRIMARY KEY (ID_F))

CREATE TABLE SHEDULES(DAY_ VARCHAR(30), ID_F INT, ID_C INT, ID_T INT, NUMBER INT)

CREATE TABLE STATEMENT(ID_ST INT, ID_C INT, ID_T INT, ID_S INT, GRADE INT, PRIMARY KEY (ID_ST))

CREATE TABLE COURSE(ID_C INT, TITLE VARCHAR(60), CREDITS INT, HOURS_ INT, PRACTICE INT, LECTURES INT, CHECKING VARCHAR(30), PRIMARY KEY(ID_C))

Якщо потрібно додати новий стовпець для таблиці, потрібно прописати наступний рядок:

ALTER TABLE (назва таблиці)

ADD (назва стовпця та його дані)

Коли виникає потреба видалити стовпець, прописується такий рядок:

ALTER TABLE (назва таблиці)

DROP COLUMN (назва стовпця)

При заповнені таблиці даними, потрібно вказати спершу які дані слід прописати, а вже після цього можна вказати значення даних. Приклад заповнення таблиці STUDENTS:

```
INSERT INTO STUDENTS (ID_S, FIRSTNAME, LASTNAME, ID_F, ID_G, MAIL, CITY, POSITION)
VALUES(14,'OLGA','MATVIIV',2,2,'OLGAMAT@GMAIL.COM','ODESA','STUDENT')
INSERT INTO STUDENTS (ID_S, FIRSTNAME, LASTNAME, ID_F, ID_G, MAIL, CITY, POSITION)
VALUES(15,'HRYSTYNA','OREH',2,2,'HRYSTYNAOR@GMAIL.COM','TERNOPIL','STUDENT')
INSERT INTO STUDENTS (ID_S, FIRSTNAME, LASTNAME, ID_F, ID_G, MAIL, CITY, POSITION)
VALUES(16,'OLENA','KOPAK',2,2,'OLENAKOP@GMAIL.COM','RIVNE','STUDENT')
INSERT INTO STUDENTS (ID_S, FIRSTNAME, LASTNAME, ID_F, ID_G, MAIL, CITY, POSITION)
VALUES(17,'MAKSYM','OLIYAR',2,2,'MAXOL@GMAIL.COM','LVIV','STUDENT')
```

Команди для наповнення таблиці GROUPS:

```
INSERT INTO GROUPS( ID_G, NAME, ID_S, ID_F)
VALUES( 3 , 'UFE' , 2 , 1 )
INSERT INTO GROUPS( ID_G, NAME, ID_S, ID_F)
VALUES( 2 , 'UFE' , 3 , 1 )
INSERT INTO GROUPS( ID_G, NAME, ID_S, ID_F)
VALUES( 4 , 'UFE' , 4 , 1 )
INSERT INTO GROUPS( ID_G, NAME, ID_S, ID_F)
VALUES( 5 , 'UFE' , 5 , 1 )
```

```
INSERT INTO GROUPS( ID_G, NAME, ID_S, ID_F)
VALUES( 6 , 'UFE' , 6 , 1 )
INSERT INTO GROUPS( ID_G, NAME, ID_S, ID_F)
VALUES( 7 , 'UFE' , 7 , 1 )
```

Для того щоб оновити дані у таблиці використовується команда UPDATE:

```
UPDATE (назва таблиці)
SET (назва стовпця = значення)
WHERE (умова)
```

Якщо потрібно видалити дані зі стовпців певної таблиці, використовується команда DELETE.

У середовищі Microsoft SQL є можливість створювати зв’язки між таблицями. Для того щоб, створити діаграму потрібно правою кнопкою миші натиснути у вкладці Object Explorer на Database Diagrams та вибрати New Database Diagram (рис. 2.2).

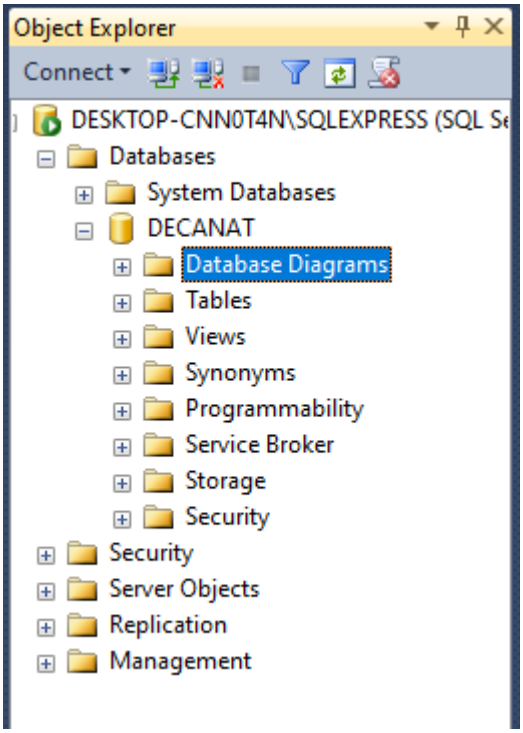


Рис.2.2. Створення діаграми у середовищі Microsoft SQL

Далі потрібно вибрати таблиці, з котрими будемо працювати (рис.2.3).

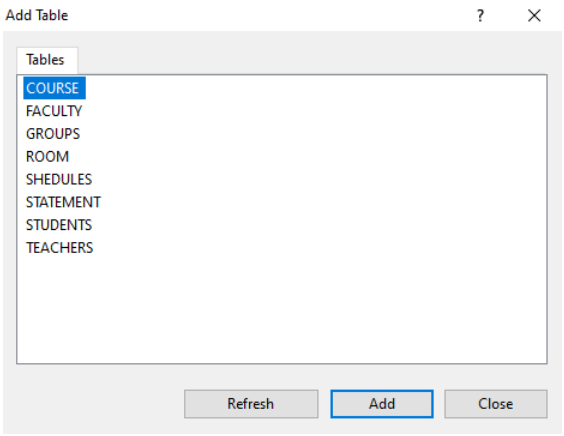


Рис.2.3. Вибір таблиць для створення діаграми

Коли відкривається робоче середовище, варто правою кнопкою миші вибрати таблицю від якої створюватиметься зв’язок з іншою таблицею та вибрати Relationships. Наступний кроком слід вказати таблицю та стовпці, котрі відповідатимуть один одному (рис.2.4).

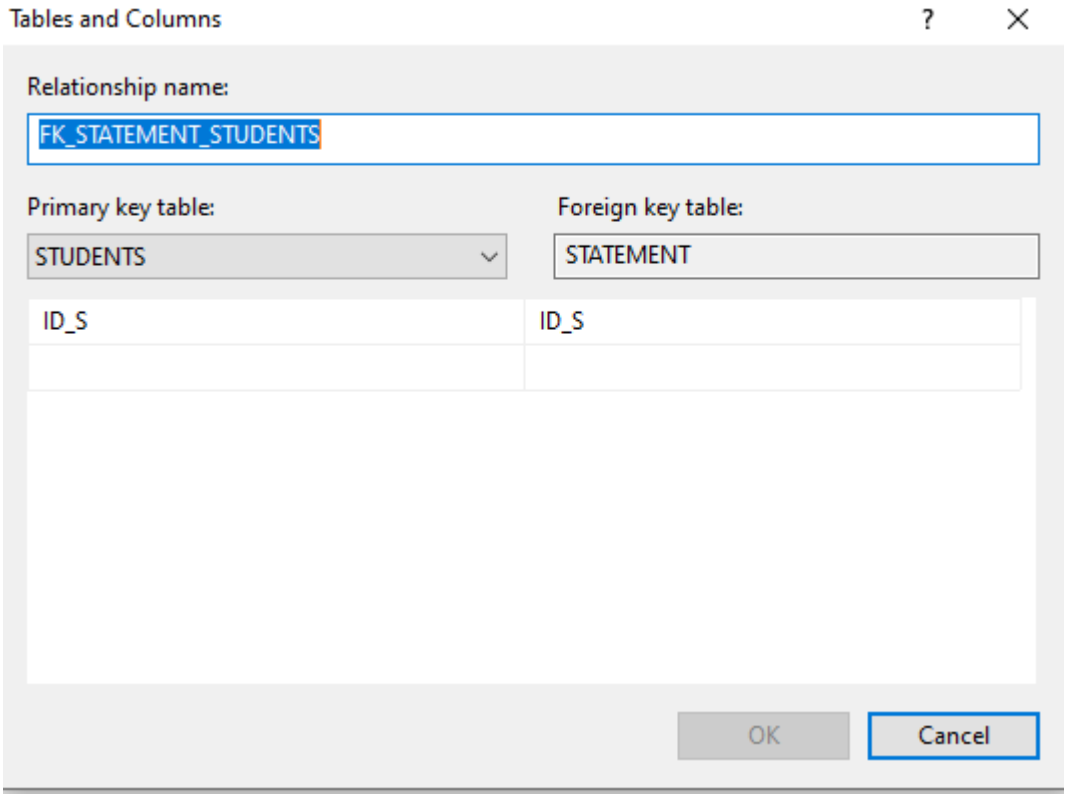


Рис. 2.4 Створення зв’язку між двома таблицями

Відповідно до створеної ER – діаграми, потрібно буде створити зв’язки між усіма таблицями.

Для інформаційної бази даних «Деканат», зв’язки між таблицями виглядають так, як зображено на рис. 2.5.

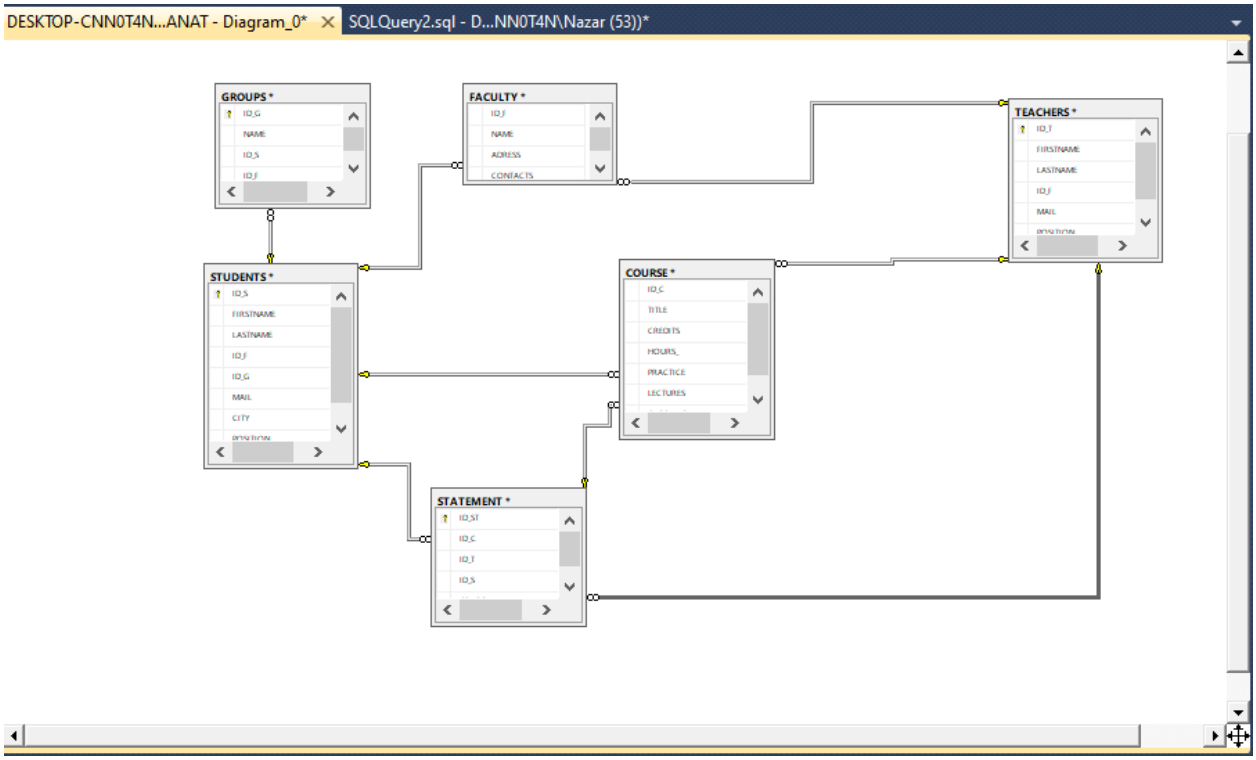


Рис. 2.5 Створенні зв’язки між таблиця у базі даних DECANAT

2.6 Створення запитів

Щоб створити запит у програмному середовищі Microsoft SQL використовується команда SELECT:

SELECT (назва стовпця)

FROM (назва таблиці)

Вибрати усі дані із таблиці FACULTY:

```
SELECT*
FROM FACULTY
```

Якщо потрібно знайти усіх студентів, котрі навчаються у групі “UFE”, прописується наступний запит:

```
SELECT STUDENTS.FIRSTNAME, STUDENTS.LASTNAME,
GROUPS.NAME
FROM STUDENTS JOIN GROUPS
```

```
ON STUDENTS.ID_S= GROUPS.ID_G
WHERE GROUPS.NAME='UFE'
```

Команда JOIN використовується для об'єднання рядків з двох або більше таблиць, на основі відповідного стовпчика між ними.

Ось різні типи JOIN у Microsoft SQL:

- (INNER) JOIN: повертає записи, що мають відповідні значення в обох таблицях;
- LEFT (OUTER) JOIN: повертає всі записи з лівої таблиці та відповідні записи з правої таблиці;
- RIGHT (OUTER) JOIN: повертає всі записи з правої таблиці та відповідні записи з лівої таблиці;
- FULL (OUTER) JOIN: повертає всі записи, коли в лівій або правій таблиці є збіг;

Приклади як працюють різні типи JOIN, зображені на рис.2.6.

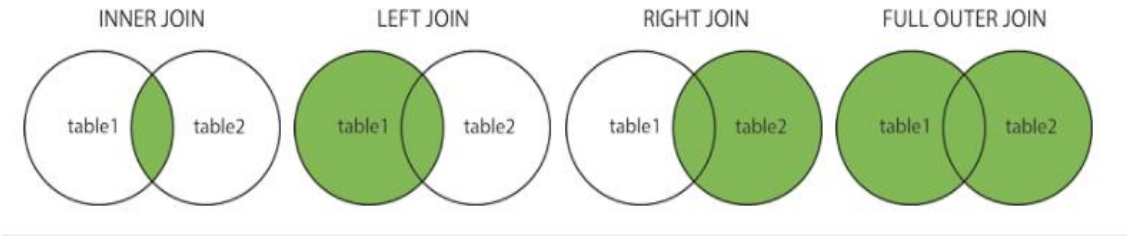


Рис.2.6 Типи JOIN

Коли потрібно знайти хто є старостою певної групи, прописується наступна команда:

```
SELECT STUDENTS.FIRSTNAME, STUDENTS.LASTNAME,
STUDENTS.POSITION
FROM STUDENTS
WHERE STUDENTS.ID_F=1
AND POSITION='STAROSTA'
```

Команда WHERE використовується для фільтрації записів. Застосовується для вилучення лише тих записів, які відповідають заданій умові. Також ця команда використовується не тільки у команді SELECT, а також у команді UPDATE та DELETE.

Для того, щоб реалізувати запит, де потрібно знайти бал певного студента з дисципліни, слід вказати таку команду:

```
SELECT STUDENTS.FIRSTNAME, COURSE.TITLE, STATEMENT.GRADE
FROM COURSE JOIN STATEMENT ON
COURSE.ID_C=STATEMENT.ID_ST
JOIN STUDENTS ON STUDENTS.ID_S=STATEMENT.ID_ST
WHERE STUDENTS.ID_S=1 AND COURSE.TITLE='ECONOMICA'
```

Команда WHEREP може бути об'єднана з AND, ORi NOT операторами.

Оператори AND і OR використовуються для фільтрації записів на основі більш ніж однієї умови:

- AND оператор відображається запис, якщо всі умови , розділені та істинні.
- OR оператор відображається записом , якщо якась з умов , розділена і може бути істиною.

NOT оператор відображає запис , якщо певна умова не відповідає дійсності.

Запит для того, щоб знайти усіх студентів, котрі живуть у Львові:

```
SELECT STUDENTS.LASTNAME, STUDENTS.CITY, GROUPS.NAME
FROM STUDENTS JOIN GROUPS ON STUDENTS.ID_S=GROUPS.ID_G
WHERE STUDENTS.CITY ='LVIV'
```

Запит, щоб дізнатися хто з викладачів викладає предмет PRAVO:

```
SELECT TEACHERS.LASTNAME, COURSE.TITLE, TEACHERS.MAIL
FROM TEACHERS JOIN COURSE ON TEACHERS.ID_T=COURSE.ID_C
WHERE COURSE.TITLE='PRAVO'
```

Запит, щоб дізнатися середній бал студентів із дисципліни PRAVO:

```
SELECT AVG(STATEMENT.GRADE) AS AVARAGE, COURSE.TITLE
```

```
FROM STUDENTS JOIN COURSE ON STUDENTS.ID_S=COURSE.ID_C
JOIN STATEMENT ON STATEMENT.ID_ST=STUDENTS.ID_S
WHERE COURSE.TITLE = 'PRAVO'
GROUP BY COURSE.TITLE
```

Види функцій:

- COUNT() - функція повертає кількість рядків , яке відповідає заданому критерію;
- AVG() - функція повертає середнє значення числового стовпця;
- SUM() - функція повертає загальну суму числового стовпця;

Запит, котрий рахує кількість студентів на факультетах:

```
SELECT COUNT(STUDENTS.ID_S), FACULTY.NAME
FROM STUDENTS JOIN FACULTY ON STUDENTS.ID_F =FACULTY.ID_F
GROUP BY FACULTY.NAME
```

Запит котрий рахує скільки студентів на певному факультеті:

```
SELECT COUNT(STUDENTS.ID_S), FACULTY.NAME
FROM STUDENTS JOIN FACULTY ON STUDENTS.ID_F =FACULTY.ID_F
WHERE FACULTY.NAME = 'EF'
GROUP BY FACULTY.NAME
```

Для того, щоб виконувалась умова, як перевести п’ятибальну систему оцінювання у алфавітну, можна використовувати команду CASE.

Оператор CASE проходить умови і повертає значення, коли дотримано першу умову (наприклад, оператор if-then-else). Отже, як тільки умова є істинною, вона перестане читати і поверне результат. Якщо жодні умови не відповідають дійсності, це повертає значення в ELSE реченні.
Якщо немає ELSEчастини і жодні умови не відповідають дійсності, вона повертає NULL.

```
SELECT STUDENTS.FIRSTNAME, STUDENTS.LASTNAME,
STATEMENT.GRADE, COURSE.TITLE,
CASE
WHEN STATEMENT.GRADE = 2 THEN 'D'
```

```
WHEN STATEMENT.GRADE = 3 THEN 'C'
WHEN STATEMENT.GRADE = 4 THEN 'B'
WHEN STATEMENT.GRADE = 5 THEN 'A'
END AS MARKS
FROM STUDENTS JOIN STATEMENT ON
STUDENTS.ID_S=STATEMENT.ID_S
JOIN COURSE ON STATEMENT.ID_C=COURSE.ID_C
```

Усі результати запитів зображено у додатку А.

ВИСНОВКИ

У ході виконання завдання було реалізовану базу даних для інформаційної системи «Деканат». В даній роботі навів короткі відомості про основні засоби, які потрібні для реалізації інформаційної системи «Деканат» і описав структури бази даних. На початковому етапі практичної частини розроблялась база даних на платформі Microsoft SQL. Наступним і завершальним етапом була побудована база даних, її наповнення, розроблення зв’язків між таблицями та створення запитів. Для вирішення проблем бази даних було проаналізовано предметну область інформаційної системи «Деканат», описані всі аспекти використання бази даних та відображено інформаційну систему за допомогою запитів.

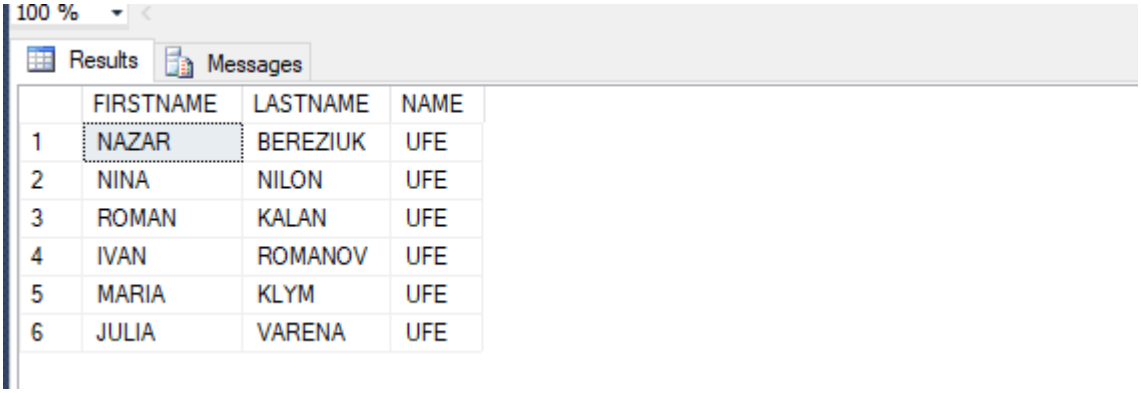
Отож, враховуючи усе вище сказане, можна стверджувати про важливу роль інформаційних системи «Деканат». Завдяки даній системі будується навчальний процес. Та полегшується робота для викладачів і студентів. Іншими словами «Деканат» створює зв’язок між студентом, викладачем та навчальним закладом, а також автоматизує певні процеси, для оперативнішої роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бази даних та інформаційні системи — Вікі ЦДПУ [Електронний ресурс]. – Режим доступу: https://cutt.ly/CyTXnR1
2. БАЗЫ ДАННЫХ - УРОК 3. РЕЛЯЦИОННЫЕ БАЗЫ ДАННЫХ [Електронний ресурс]. – Режим доступу: https://site-do.ru/db/db3.php
3. ОСНОВИ БАЗ ДАНИХ [Електронний ресурс]. – Режим доступу: http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/118/118.html
4. ЧТО ТАКОЕ СУБД - RU-CENTER [Електронний ресурс]. – Режим доступу: https://www.nic.ru/help/chto-takoe-subd_8580.html
5. РОЗДІЛ 1 . МОДЕЛЮВАННЯ РЕЛЯЦІЙНОЇ СТРУКТУРИ БД [Електронний ресурс]. – Режим доступу: https://studfile.net/preview/5391756/
6. КОРОТКЕ ОЗНАЙОМЛЕННЯ З ІНСТРУМЕНТОМ ДЛЯ БАЗ ДАНИХ – PHPMYADMIN | SEBWEO [Електронний ресурс]. – Режим доступу: https://sebweo.com/korotke-oznajomlennya-z-instrumentom-dlya-baz-danih-phpmyadmin/
7. OPEN SERVER PANEL — ЛОКАЛЬНЫЙ ВЕБ-СЕРВЕР ДЛЯ WINDOWS. СКАЧАТЬ HTTP WAMP ВЕБ СЕРВЕР. [Електронний ресурс]. – Режим доступу: https://ospanel.io/
8. НОРМАЛІЗАЦІЯ ВІДНОШЕНЬ [Електронний ресурс]. – Режим доступу: https://life-prog.ru/ukr/1_331_normalizatsiya-vidnoshen.html
9. НОРМАЛІЗАЦІЯ БАЗ ДАНИХ — ВІКІПЕДІЯ [Електронний ресурс]. – Режим доступу: https://cutt.ly/tyTCqT8
10. ХОТОРН Р. РАЗРАБОТКА БАЗ ДАННЫХ MICROSOFT SQL SERVER 2000 НА ПРИМЕРАХ: АНГЛ.- М.: ВИЛЬЯМС, 2001.- 464 С.- ISBN 5-8459-0187-1. УДК – 004.65
11. W3SCHOOLS ONLINE WEB TUTORIALS [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/

12. ГАРСИА-МОЛИНА Г., УЛЬМАН Д., УИДОМ Д. СИСТЕМЫ БАЗ ДАННЫХ. ПОЛНЫЙ КУРС.: ПЕР. С АНГЛ. - М.: ИЗДАТЕЛЬСКИЙ ДОМ "ВИЛЬЯМС", 2004. - 1088 С.
13. ДЕЙТ К. ДЖ. ВВЕДЕНИЕ В СИСТЕМЫ БАЗ ДАННЫХ. :ПЕР. С АНГЛ. – 6-Е ИЗД. – К.: ДИАЛЕКТИКА, 1998. – 784 С.
14. КАЛЯНОВ Г.Н. CASE-ТЕХНОЛОГИИ. КОНСАЛТИНГ В АВТОМАТИЗАЦИИ БИЗНЕС-ПРОЦЕССОВ. – 3-Е ИЗД. – М.: ГОРЯЧАЯ ЛИНИЯ-ТЕЛЕКОМ,2002. - 320 С.
15. КАРПОВА Т.С. БАЗЫ ДАННЫХ: МОДЕЛИ, РАЗРАБОТКА, РЕАЛИЗАЦИЯ. – СПБ.: ПИТЕР, 2001. – 304 С.
16. КОГАЛОВСКИЙ М.Р. ЭНЦИКЛОПЕДИЯ ТЕХНОЛОГИЙ БАЗ ДАННЫХ. – М.: ФИНАНСЫ И СТАТИСТИКА, 2002. – 800 С.
17. КОННОЛИ Т., БЕГГ К., СТРАЧАН А. БАЗЫ ДАННЫХ: ПРОЕКТИРОВАНИЕ, РЕАЛИЗАЦИЯ И СОПРОВОЖДЕНИЕ. ТЕОРИЯ И ПРАКТИКА., 2-Е ИЗД.: ПЕР. С АНГЛ. – М.: ИЗДАТЕЛЬСКИЙ ДОМ «ВИЛЬЯМС», 2001. – 1120 С.

Результат запиту, де показує усіх студентів групи УФЕ.



100 %

Results Messages

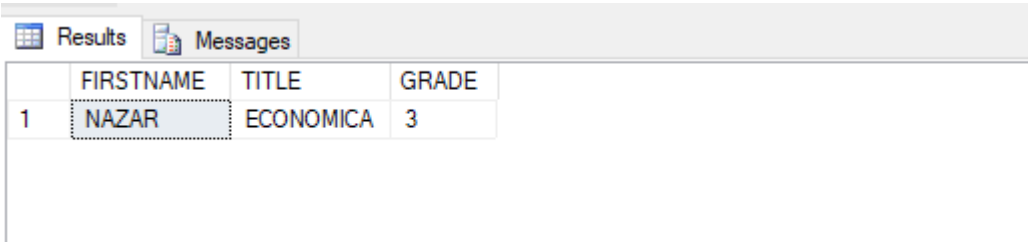
	FIRSTNAME	LASTNAME	NAME
1	NAZAR	BEREZIUK	UFE
2	NINA	NILON	UFE
3	ROMAN	KALAN	UFE
4	IVAN	ROMANOV	UFE
5	MARIA	KLYM	UFE
6	JULIA	VARENA	UFE

Результат запиту, де показує старосту групи УФЕ.

Results Messages

	FIRSTNAME	LASTNAME	POSITION
1	JULIA	VARENA	STAROSTA

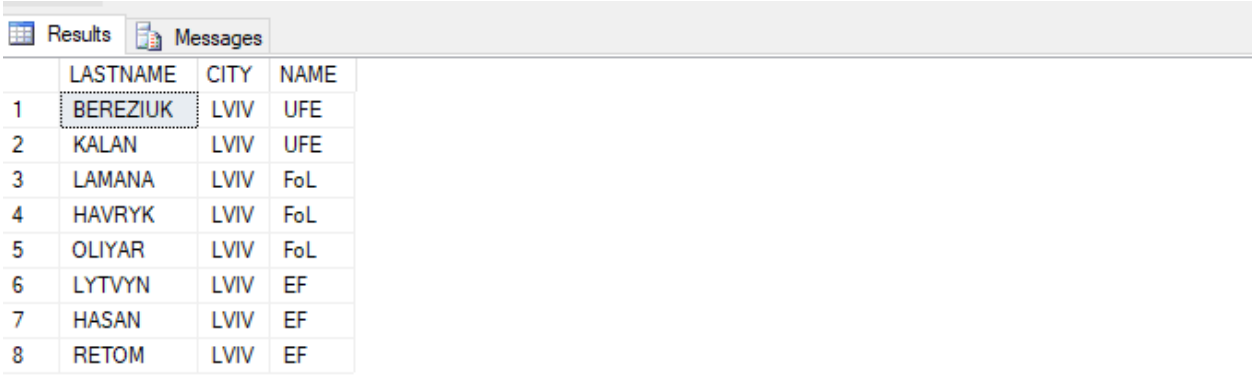
Результат запиту, де показує оцінку певного студента.



Results Messages

	FIRSTNAME	TITLE	GRADE
1	NAZAR	ECONOMICA	3

Результат запиту, де показує студентів котрі проживають у Львові.



Results Messages

	LASTNAME	CITY	NAME
1	BEREZIUK	LVIV	UFE
2	KALAN	LVIV	UFE
3	LAMANA	LVIV	FoL
4	HAVRYK	LVIV	FoL
5	OLIYAR	LVIV	FoL
6	LYTVYN	LVIV	EF
7	HASAN	LVIV	EF
8	RETOM	LVIV	EF

Результат запиту, де показує викладача та його контактні дані, який викладає певну дисципліну .

Results Messages

	LASTNAME	TITLE	MAIL
1	KORODUN	PRAVO	LEODIS.KORO@GMAIL.COM

Середній бал студента з певної дисципліни.

	AVARAGE	TITLE
1	4	PRAVO

Кількість студентів на факультетах.

Results Messages

	(No column name)	NAME
1	12	EF
2	9	FACULTYofLOW
3	8	FUFB

Перетворення п’ятибальної системи у алфавітну.

Results Messages

	FIRSTNAME	LASTNAME	GRADE	TITLE	MARKS
1	NAZAR	BEREZIUK	3	ECONOMICA	C
2	VALENTINA	KACHMAR	4	PRAVO	B
3	OLEG	VASIUTA	5	BUSSINESS IN SMM	A
4	BOGDAN	MASLOW	2	BUSSINESS ENGLISH	D
5	ROSTYSLAV	MAK	4	HISTORY	B
6	OLGA	MATVIIV	2	ADMIN OF BD	D
7	ROMANA	LAMANA	5	1C	A
8	ROMANA	LAMANA	3	BUSSINESS IN SMM	C
9	JULIA	VARENA	2	1C	D
10	VITALII	RETOM	5	HISTORY OF LAW	A
11	BOGDAN	MASLOW	2	UKRAIANAIN LAW	D
12	ANDRII	PRON	4	BUSSINESS IN SMM	B
13	NAZAR	BEREZIUK	3	ECONOMICA	C
14	NAZAR	BEREZIUK	3	BUSSINESS IN SMM	C
15	NAZAR	BEREZIUK	5	UKR LANGUAGE	A
16	NAZAR	BEREZIUK	5	HISTORY	A
17	NAZAR	BEREZIUK	3	ADMIN OF BD	C
18	NAZAR	BEREZIUK	4	BUSSINESS ENGLISH	B
19	NAZAR	BEREZIUK	3	STRAHUVANYA	C
20	NAZAR	BEREZIUK	5	1C	A
21	VALENTINA	KACHMAR	5	BUSSINESS IN SMM	A
22	VALENTINA	KACHMAR	4	UKR LANGUAGE	B

Вибір усіх стовпців із таблиці FACULTY.

Results Messages

	ID_F	NAME	ADRESS	CONTACTS
1	1	FUFB	KOPERNIKA,3	232-80-85
2	2	FACULTYofLOW	STUDENTSKA	232-80-86
3	3	EF	PR.SVOBODY	232-80-87

Кількість студентів на певному факультеті

	(No column name)	NAME
1	12	EF