

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**Український державний університет науки і технологій**

Кафедра «Комп’ютерні інформаційні технології»

**Курсовий проект**

**з дисципліни «Бази даних»**

Виконав:

Студент гр. ПЗ2011

Проценко Роман

Прийняв:

Самойлов С. П.

Дніпро

2023

**Міністерство освіти і науки України**

**Український державний університет науки і технологій**

Факультет: Комп’ютерних технологій і систем

Кафедра: Комп’ютерні інформаційні технології

Спеціальність: 121. Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ  
Зав. кафедри «КІТ»  
доцент Вадим ГОРЯЧКІН  
« 28» грудня 2023р.

**ЗАВДАННЯ**до курсового проекту по дисципліні **«Бази даних»**

студента \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Проценка Романа Олександровича\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема проекту: Розробка реляційної бази даних, проектування структури сховища даних. Проектування журналу успішності студентів.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Дата видачі завдання \_28 вересня 2023 р.\_\_\_\_\_\_\_\_\_\_\_\_\_
3. Перелік питань до розробки:  
   Вибір предметної області БД, інформаційно-логічне і концептуальне проектування бази даних у рамках EER моделі.
4. Постановка завдання, вибір предметної області, специфікація вимог до управління даними, визначення основних типів запитів.
   1. Розробка інформаційно-логічної і концептуальної EER моделі БД.
   2. Визначення типів сутностей, атрибутів, характеристик зв’язків, встановлення можливих і первинних ключів.
   3. Аналіз вимог до БД, виконання процедур спеціалізації/генерації, категоризації типів сутностей EER моделі БД.
5. Розробку логічного проекту бази даних.
   1. Розробка попередньої логічної схеми БД на основі EER моделі.
   2. Розподіл не ключових атрибутів, нормалізація БД.
   3. Вилучення зайвих, складних (m : n) і рекурсивних зв’язків, а також зв’язків що мають атрибути.
   4. Забезпечення цілісності БД.
   5. Аналіз запитів, утворення карт транзакцій.
   6. Заключна EER модель БД.
6. Розробка фізичного проекту бази даних.
   1. Розробка структур таблиць.
   2. Розробка схеми БД з урахуванням можливостей СУБД.
   3. Розробка запитів і форм.
   4. Контрольні випробування із керування БД, реалізація запитів (відповідність картам транзакцій).
7. Проектування логічної структури реляційних аналітичних систем накопичення і оброблення даних – ROLAP.
   1. Вибір предметної області, дослідження вимог до сховища даних складання попереднього набору аналітичних запитів.
   2. Розробка моделі сховища – вибір мір, вимірів, таблиць фактів.
   3. Графічні моделі нормалізованих (сніжинка) і ненормалізованих (зірка) схем сховища, які ураховують набори аналітичних запитів.
   4. Графічне представлення двох аналітичних запитів, а також реалізація запитів засобами OLAP для СУБД MS SQL Server.
8. Висновок по проекту – відмінності та особливості розробки інформаційних систем баз даних і сховищ даних.

Термін виконання курсового проекту **21 грудня 2023 р**.

Керівник курсового проекту\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_Самойлов С. П.\_\_\_/

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_Проценко Р.О.\_\_\_/

**Календарний план**

|  |  |  |  |
| --- | --- | --- | --- |
| № | Етапи курсового проекту | Строк виконання | Примітка |
| 1 | Отримання завдання | 28.09.2023 |  |
| 2 | Аналіз методів вирішення поставленої задачі | 14.10.2023 |  |
| 3 | Розробка концептуальної моделі | 21.10.2023 |  |
| 4 | Логічне проектування бази даних | 28.10.2023 |  |
|  | Фізичне проектування бази даних |  |  |
|  | Нормалізація функціональних відносин в базі даних |  |  |
| 6 | Розробка клієнтського додатку | 14.11.2023 |  |
|  | Розробка інтерфейсу користувача | 21.11.2023 |  |
| 7 | Розробка сховища даних | 28.11.2023 |  |
| 8 | Розробка запитів до сховища даних | 07.11.2023 |  |
| 9 | Оформлення пояснювальної записки | 14.12.2023 |  |
| 10 | Здача пояснювальної записки | 21.12.2023 |  |
| 11 | Захист курсового проекту | 21.12.2023 |  |

**Реферат**

Курсовий проект: 36 стр., 8 рис., 7 джерел літератури, 3 додатки.

Об’єкт дослідження – база даних студентів.

Ціль роботи - розробка реляційної бази даних, проектування структури сховища даних.

Метод дослідження – Розробка бази даних на основі реляційної моделі представлення даних.

Було проведено аналіз вибраної галузі, ідентифіковано ключові моделі та зв'язки. На основі цього дослідження була створена концептуальна модель бази даних для освітньої галузі, а також розроблена реляційна модель бази даних з використанням сервера MS SQL Server. Крім того, був створений клієнт бази даних на мові C# з використанням Windows Forms.

БАЗИ ДАНИХ, РЕЛЯЦІЙНА МОДЕЛЬ, БАЗА ДАНИХ СТУДЕНТІВ, MICROSOFT SQL SERVER, WINDOWS FORMS.

**Зміст**

Розробка системи управління БД

[Опис предметної області 6](#_Toc154668621)

[Обґрунтування вибору СУБД 7](#_Toc154668621)

[Розробка концептуальної моделі бази даних 8](#_Toc154668621)

[Визначення сутностей та їх елементів 9](#_Toc154668621)

[Побудова реляційной моделі БД 11](#_Toc154668621)

[Нормалізація бази даних 1](#_Toc154668621)2

[Схема бази даних 1](#_Toc154668621)4

[Розробка проекту БД 1](#_Toc154668621)6

[Сховище даних з архутектурою «зірка» 18](#_Toc154668622)

[Сховище даних з архутектурою «сніжинка» 20](#_Toc154668623)

[**Висновки** 2](#_Toc154668624)1

**[Додаток 1](#_Toc154668624)** [2](#_Toc154668624)2

**[Додаток 2](#_Toc154668624)** [26](#_Toc154668624)

[**Додаток 3** 34](#_Toc154668625)

**Опис предметної області**

Предметною областю даного проекту є система управління інформацією про студентів, що дозволяє зберігати, обробляти та переглядати дані щодо студентів.

## Постановка завдання

Розробити об'єктно-орієнтовану програму для управління інформацією бази даних студентів:

* ***Можливість відбору даних за вказаними критеріями:***

Реалізувати функціонал для відбору студентів даних відповідно до заданих критеріїв і т.д.

* ***Пошук необхідної інформації в базі даних:***

Створити можливість здійснення пошуку в базі даних для виявлення конкретної інформації про студентів.

* ***Захист бази даних від несанкціонованого доступу:***

Реалізувати механізми захисту бази даних від неправомірних змін та забезпечити реєстрацію користувачів для контролю доступу.

* ***Можливість додавання потрібної інформації в базу даних:***

Розробити функціонал для додавання нових даних про студентів в базу даних, таких як оцінки, встигаємість.

* ***Інтерфейс користувача з перевіркою правильності введених даних:***

Створити зручний інтерфейс користувача з механізмами перевірки введених даних для уникнення помилок та забезпечення консистентності інформації в базі даних.

При цьому слід враховувати ефективність взаємодії програми з базою даних, забезпечуючи безпеку обробки і збереження інформації.

**Обґрунтування вибору СУБД**

Вибір Системи Управління Базами Даних (СУБД) — це стратегічне рішення, що визначає ефективність та надійність роботи з інформацією в проекті. Вибір Microsoft SQL Server для використання в системі, матиме наступні переваги:

***Широкі функціональні можливості:*** Microsoft SQL Server володіє розширеним функціоналом, який включає в себе різноманітні можливості управління даними, забезпечуючи високий рівень функціональності для розробників та адміністраторів баз даних.

***Інтеграція з іншими продуктами Microsoft:*** Microsoft SQL Server ефективно взаємодіє з іншими продуктами компанії Microsoft, такими як .NET Framework, Azure, Excel тощо, що робить його ідеальним вибором для екосистеми Microsoft.

***Зручний інтерфейс користувача:*** SQL Server Management Studio (SSMS) надає зручний інтерфейс для адміністрування та роботи з базами даних. Його інтуїтивно зрозумілий інтерфейс полегшує розробку, оптимізацію та моніторинг баз даних.

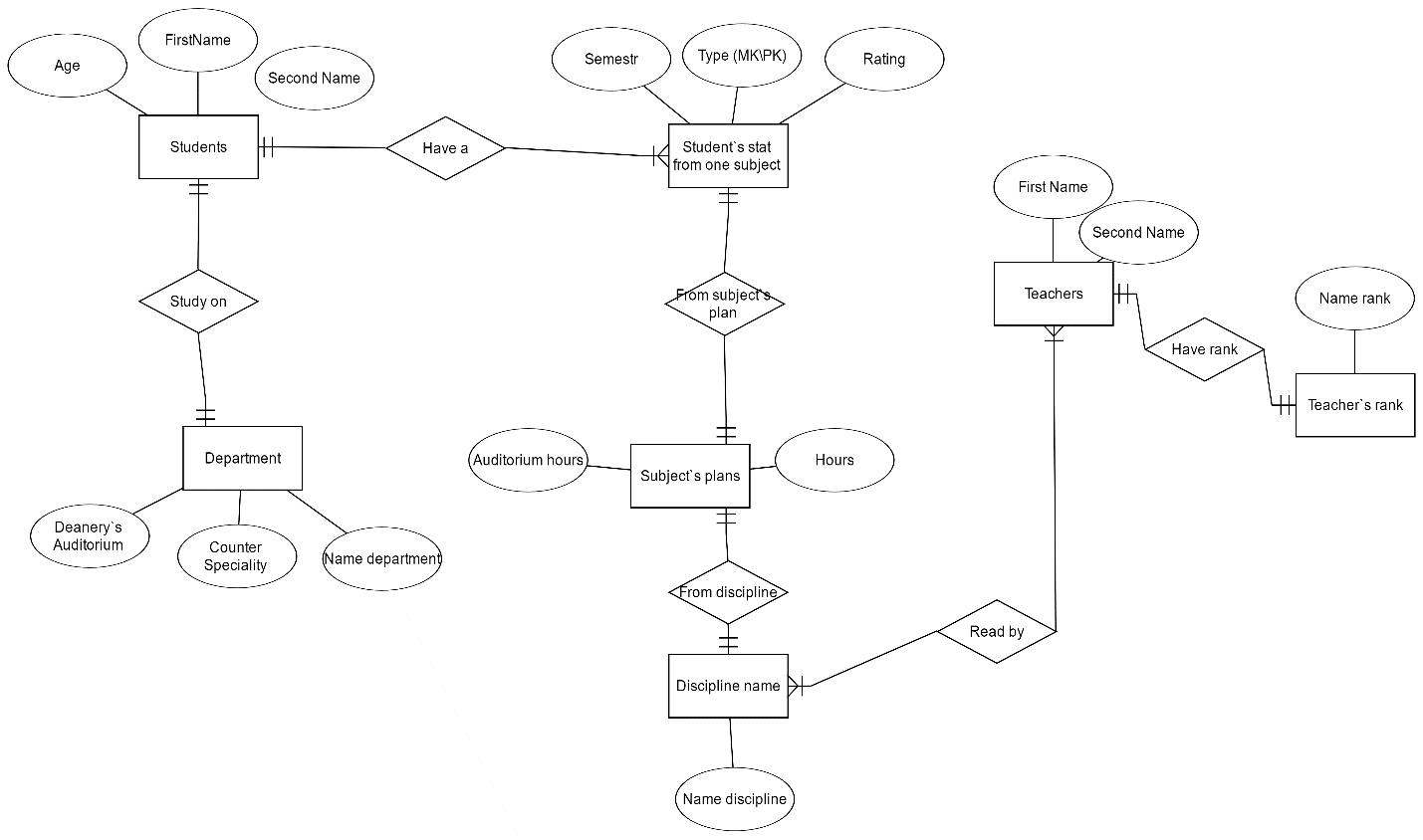
***Підтримка транзакцій і безпеки:*** Microsoft SQL Server надає ефективні механізми для керування транзакціями, забезпечуючи цілісність даних. Додатково, він має високі стандарти безпеки, такі як ролева аутентифікація та засоби шифрування.

***Масштабованість і продуктивність:*** SQL Server здатний ефективно масштабуватися від невеликих проектів до великих підприємств, а також надає інструменти для оптимізації продуктивності.

Обравши Microsoft SQL Server, вибираєте не лише потужний інструмент для роботи з базами даних, а й інтегровану екосистему, яка спрощує розробку та підтримку проекту, забезпечуючи високий стандарт ефективності та надійності.

**Розробка концептуальної моделі бази даних**

Сутнісно-реляційна діаграма — це графічний інструмент для моделювання взаємозв'язків між різними сутностями (об'єктами) в системі або додатку. Вона базується на сутнісно-реляційному підході до проектування баз даних. Ця діаграма використовується для відображення структури бази даних та взаємозв'язків між різними сутностями.



Мал. 1. ER-діаграма бази даних моделі предметної області

**Визначення сутностей та їх елементів**

|  |  |
| --- | --- |
| **Сутність** | **Опис** |
| Students | Містить головну інформацію про студентів.  C:\Users\Storchak\Desktop\Снимок.JPG |
| Student`s stat | Містить інформацію про успішність та заборгованість студентів по кожному предмету.  C:\Users\Storchak\Desktop\Снимок.JPG |
| Teachers | Містить головну інформацію про викладачів  C:\Users\Storchak\Desktop\Снимок.JPG |
| Teacher\_ranks | Містить інформацію про ступінь викладачів  C:\Users\Storchak\Desktop\Снимок.JPG |

|  |  |
| --- | --- |
| Disciplines | Містить головну інформацію про навчальні предмети  C:\Users\Storchak\Desktop\Снимок.JPG |
| TeachersSubject | Містить інформацію про предмети які читають викладачі.  C:\Users\Storchak\Desktop\Снимок.JPG |
| TypeSubject | Містить інформацію про тип предмета у рейтингу студентів.  C:\Users\Storchak\Desktop\Снимок.JPG |
| Subject\_plan | Містить інформацію план навчальних предметів.  C:\Users\Storchak\Desktop\Снимок.JPG |
| Departments | Містить інформацію про факультети вищих навчальних закладів.  C:\Users\Storchak\Desktop\Снимок.JPG |
| Speciality | Містить інформацію про спеціальності що є на факультетах.  C:\Users\Storchak\Desktop\Снимок.JPG |

**Побудова реляційной моделі БД**

Реляційний метод побудови баз даних визначається своєрідною ідеологією, що спрощує структуру даних та сприяє їх зручному використанню. Його основні принципи включають табличне подання даних та використання формального апарату алгебри відносин та реляційного числення для обробки інформації.

У реляційній моделі дані організовані у вигляді двовимірних таблиць, де кожен елемент представляє собою один елемент даних. Властивості реляційної таблиці включають однорідність стовпців, унікальні імена стовпців і можливість зміни порядку рядків і стовпців.

Відносини в реляційній моделі представлені у вигляді таблиць, де рядки відповідають записам, а стовпці — атрибутам. Ключі, такі як первинний ключ, грають важливу роль у визначенні унікальності записів.

Один з головних аспектів реляційної моделі — це можливість встановлення відносин між таблицями за допомогою ключів. Це забезпечує зв'язаність інформації та робить модель гнучкою для подальших розширень.

Хоча реляційний підхід є поширеним і забезпечує зручний доступ до даних, він також має свої недоліки, включаючи обмежену швидкість доступу, складність проектування та можливість зайвого розширення бази даних через нормалізацію. Тим не менше, цей підхід залишається одним із найпоширеніших завдяки своїй простоті та математичній базі.

**Нормалізація бази даних**

Нормалізація бази даних — це процес організації структури даних з метою мінімізації дублювання і підвищення ефективності обробки та оновлення інформації. Вона дозволяє визначити оптимальний набір відносин (таблиць) для зберігання даних, щоб спростити взаємодію з базою та забезпечити надійність і консистентність даних.

Організація атрибутів у відносинах повинна бути раціональною, що передбачає уникнення зайвого або непотрібного зберігання інформації. Мета полягає в тому, щоб кожна таблиця відповідала певним вимогам нормалізації відносин та могла ефективно опрацьовувати дані.

Існує п'ять рівнів нормальних форм (1НФ, 2НФ, 3НФ, BCNF, 4НФ), проте для більшості ситуацій найчастіше використовуються лише три: перша, друга і третя нормальні форми. Кожен рівень нормалізації має свої критерії, які визначаються відносинами між атрибутами таблиць.

Важливо враховувати, що хоча нормалізація сприяє оптимізації бази даних, в деяких випадках може виникнути потреба у компромісі між оптимізацією та зручністю використання даних, особливо в великих і складних системах.

**Перша нормальна форма**

Перша нормальна форма (1НФ) — це стан відношення в реляційній базі даних, при якому всі атрибути цього відношення мають прості (атомарні) значення. Іншими словами, значення в кожному атрибуті не можуть бути ні списками, ні множинами складних значень. Умова атомарності гарантує відсутність повторюваних груп у відношенні, що визначає його як нормалізоване.

В першій нормальній формі кожна схема відносини повинна бути атомарною, тобто не допускати деталізації значень. Це важливо для забезпечення чіткості та однозначності представлення даних у реляційній базі.

Інша важлива точка — це умова, що схема всієї бази даних також повинна бути в 1НФ, оскільки кожна окрема схема відношення має відповідати цій нормальній формі. Такий підхід сприяє консистентності та ефективній організації бази даних. Щоб досягти 1НФ, можуть використовуватися методи нормалізації, що включають унікалізацію та розщеплення відносин.

**Друга нормальна форма**

Друга нормальна форма (2НФ) застосовується до відносин, у яких первинний ключ складається зі складових (композитних) атрибутів, тобто відносин, у яких ключ складається з двох чи більше атрибутів. Важливим аспектом є те, що вся кожна компонента ключа повинна функціонально визначати всі інші атрибути відносини. Відносина, у якої первинний ключ складається з одного атрибута, завжди вважається вже в другій нормальній формі.

**Третя нормальна форма**

Третя нормальна форма (3НФ) вимагає, щоб кожен неключовий атрибут відносини функціонально залежав тільки від первинного ключа, а не від інших неключових атрибутів. Іншими словами, кожен атрибут, що не входить до складу первинного ключа, повинен залежати від ключа, а не від інших неключових атрибутів.

Основна ідея 2НФ та 3НФ полягає в уникненні аномалій та забезпеченні логічної структури бази даних через поділ на окремі таблиці та уникатові ключі. Це сприяє консистентності та оптимізації бази даних для більш ефективного управління інформацією.

**Четверта нормальна форма**

Четверта нормальна форма (4НФ) вводиться для уникнення многозначних залежностей, які можуть виникнути в відносинах. Це досягається шляхом розбиття таблиць для збереження неконтекстуальних (многозначних) атрибутів.

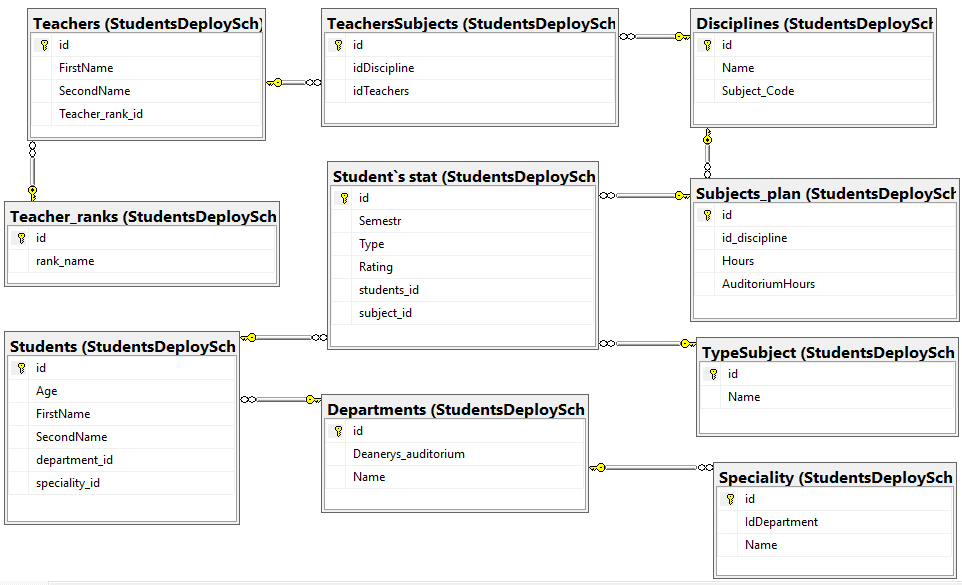
**П’ята нормальна форма**

П'ята нормальна форма (5НФ) вимагає розбиття відносин для уникнення ситуацій, коли існують залежності відносин між неключовими атрибутами, що можуть призвести до аномалій та неконсистентності даних.

Враховуючи 4НФ та 5НФ, база даних досягає високого рівня структурованості та уніфікованості, забезпечуючи ефективне управління інформацією та уникнення аномалій в даних. Ці нормальні форми доповнюють попередні, сприяючи оптимізації та консистентності бази даних.

**Схема бази даних**

База даних розроблена в проекті має на меті ефективно зберігати та управляти інформацією про студентів та пов'язаних з ними даними. Основні компоненти структури бази даних відображають високий рівень нормалізації, що сприяє уникненню дублювання даних та забезпечує консистентність даних.



Мал. 2. База даних нормалізовна до 3 форми.

У нашій базі даних всі таблиці відповідають першій нормальній формі, оскільки всі атрибути є простими та не допускають повторюваних груп.

Згідно з визначенням другої нормальної форми, відношення перебуває в цій нормальній формі, якщо воно вже знаходиться в першій нормальній формі, і кожен атрибут, що не є ключем, повністю залежить від цілого первинного ключа, а не лише від частини його. У нашій базі даних такий підхід врахований, і всі атрибути, не є ключовими, повністю залежать від цілого первинного ключа відповідної таблиці.

Щодо третьої нормальної форми, вона передбачає відсутність транзитивних залежностей відносно первинного ключа. Аналізуючи наші таблиці, ми підтверджуємо відсутність таких транзитивних залежностей, що підтверджує, що всі таблиці бази даних знаходяться в третій нормальній формі.

**Розробка проекту БД**

На етапі фізичного проектування бази даних, виходячи з розробленої логічної моделі, формується фізична модель, яка враховує особливості конкретної системи управління базами даних (СУБД). Основною метою цього етапу є конкретизація та оптимізація структури даних для забезпечення оптимальної продуктивності та ефективного взаємодії з системою.

На початковому етапі фізичного проектування створюються основні таблиці, які відображають структуру даних, визначену на логічному етапі. Після цього додаються зовнішні ключі, які встановлюють зв'язки між різними таблицями, що відображають взаємозв'язки між об'єктами домену.

Логічна модель даних залежить від вибраної моделі організації даних, такої як реляційна чи мережева. У той час як логічна модель є абстракцією, фізична модель докладно описує, як саме дані будуть зберігатися та оброблятися в межах конкретної СУБД.

Необхідність фізичного проектування випливає із специфіки обраної СУБД. На цьому етапі приймаються рішення щодо створення реляційних таблиць, встановлення обмежень, визначення методів зберігання даних та розробки засобів захисту інформації.

Важливо відзначити, що між логічним та фізичним проектуванням існує нерозривний зворотний зв'язок. Зміни, внесені на етапі фізичного проектування для оптимізації продуктивності, можуть впливати на структуру логічної моделі даних.

Microsoft SQL Server є комерційною системою керування базами даних, що використовуватиметься в проекті. Вона підтримує мову запитів Transact-SQL, яка є реалізацією стандарту ANSI/ISO щодо SQL з розширеннями. SQL Server може бути успішно використаний для роботи як з невеликими і середніми за розміром базами даних, так і з великими базами даних масштабу підприємства. Його конкурентоспроможність на ринку баз даних вже багато років підтверджується успішною конкуренцією з іншими системами управління базами даних.

*Скрипт розгортання бази даних наведено в додатку 1.*

**Транзакції та забезпечення цілісності**

**Транзакції**

Транзакція в базі даних (БД) представляє собою логічний блок операцій, які виконуються як єдине ціле, при цьому вони вважаються атомарними і виконуються або повністю, або не виконуються взагалі. Транзакції використовуються для забезпечення консистентності та цілісності даних в базі даних.

Основні властивості транзакцій включають такі концепції:

***Атомарність (Atomicity):*** Транзакція виконується або повністю, або не виконується взагалі. Це означає, що якщо одна операція транзакції не вдалася, то всі інші операції також скасовуються, і база даних повертається до початкового стану.

***Консистентність (Consistency):*** Транзакція переводить базу даних з одного стабільного стану в інший, забезпечуючи цілісність даних. Якщо транзакція порушує будь-які правила цілісності, вона скасовується, і база даних залишається незмінною.

***Ізоляція (Isolation):*** Транзакції виконуються паралельно, і їх вплив на одна одну має бути мінімізованим. Це забезпечує унікальність результату транзакції для кожного користувача.

***Тривалість (Durability):*** Після успішного завершення транзакції її зміни залишаються в базі даних і витримують будь-які відмови системи.

Транзакції дозволяють забезпечити надійність та коректність взаємодії з базою даних, що є критичним для забезпечення правильності та надійності операцій з даними.

**Забезпечення цілісності бази даних**

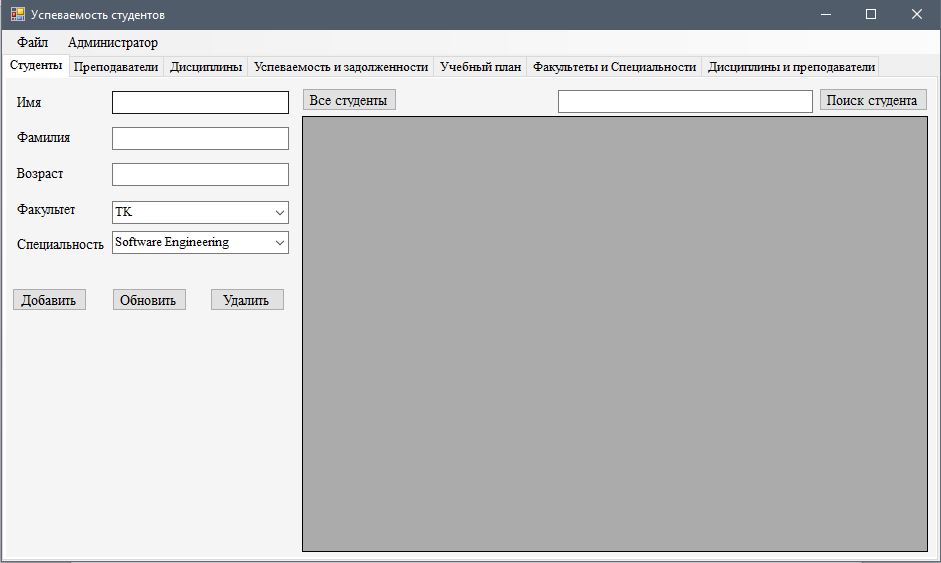
Цілісність даних у базі даних визначається її здатністю забезпечувати коректність та валідність даних на будь-якому етапі їхнього зберігання та обробки. Цей аспект критично важливий для забезпечення надійності та правильності операцій з інформацією в базі даних. Обмеження відіграють ключову роль у підтримці цілісності, визначаючи умови, які повинні виконуватися для вставки або модифікації даних. Розглянемо більше інформації про деякі типи обмежень:

* ***NOT NULL (Не порожнє):*** Це обмеження використовується для того, щоб гарантувати, що в значеннях конкретного стовпця немає порожніх (NULL) значень. Воно особливо корисне для тих стовпців, де необхідно мати обов'язкові значення, такі як ідентифікатори чи обов'язкові атрибути.
* ***UNIQUE (Унікальне):*** Обмеження унікальності визначає, що всі значення в певному стовпці чи групі стовпців повинні бути унікальними в межах таблиці. Це дозволяє уникнути дублювання даних та встановити унікальні ідентифікатори.
* ***PRIMARY KEY (Первинний ключ):*** Це обмеження визначає унікальний ідентифікатор для кожного запису в таблиці. Первинний ключ може складатися з одного чи декількох стовпців і використовується для унікальної ідентифікації записів та встановлення відносин між таблицями.
* ***FOREIGN KEY (Зовнішній ключ):*** Це обмеження вказує на зв'язок між двома таблицями. Зовнішній ключ визначається в таблиці, яка посилається на значення у первинному ключі іншої таблиці. Це забезпечує цілісність даних та забороняє вставку значень, які не мають відповідного зв'язку в батьківській таблиці.
* ***CHECK (Перевірка):*** Обмеження CHECK дозволяє визначити умови, які повинні виконуватися для значень в певному стовпці. Це може бути використано для перевірки, чи відповідають значення певним логічним умовам чи обмеженням.

Обмеження грають важливу роль у забезпеченні консистентності та цілісності даних в базі даних, а також сприяють уникненню помилок та некоректних операцій. Їх використання дозволяє зробити базу даних більш стійкою до невірних або некоректних дій користувачів або програм.

**Розробка форм та запитів до бази даних**

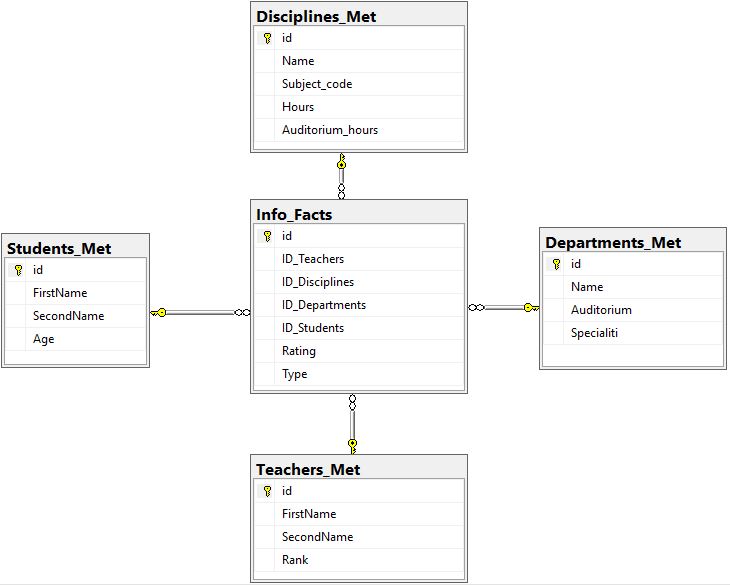
Привабливий графічний інтерфейс користувача (GUI) є важливим елементом для успішності будь-якого додатка чи системи. Не лише забезпечує зручність використання та ефективну взаємодію, але й привертає увагу та створює позитивне враження. Гарно спроектований GUI полегшує навчання користувачів, сприяє виконанню завдань та зменшує кількість помилок. В розробці форм та запитів до бази даних, зрозумілі форми та інтуїтивно зрозумілі елементи керування полегшують введення та отримання даних.



## Сховище даних з архутектурою «зірка»

Схема «зірки» - спеціальна організація реляційних таблиць, зручна для зберігання багатовимірних показників.

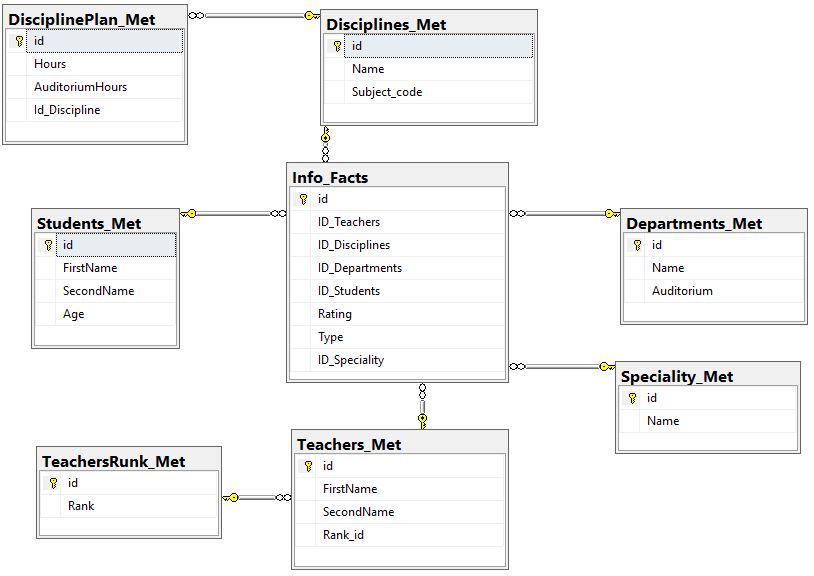
Модель даних складається з двох типів таблиць: однієї таблиці фактів - центр «зірки» - і декількох таблиць вимірів.



*Мал. 3. Сховище з архутектурою “Зірка”*

## Сховище даних з архутектурою «сніжинка»

Схема сніжинки отримала свою назву за свою форму, у вигляді якої відображається логічна схема таблиць в багатовимірної базі даних. Так само як і в схемі зірки, схема сніжинки представлена ​​централізованої таблицею фактів, з'єднаної з таблицями вимірювань. Відмінністю є те, що тут таблиці вимірювань нормалізовані з рядом інших пов'язаних вимірювальних таблиць, - в той час як в схемі зірки таблиці вимірювань повністю денормалізовани, з кожним виміром представленим у вигляді єдиної таблиці, без з'єднань на пов'язані таблиці в схемі сніжинки. Чим більше ступінь нормалізації таблиць вимірів, тим складніше виглядає структура схеми сніжинки. Створюваний «ефект сніжинки» зачіпає тільки таблиці вимірювань, і не застосовується для таблиць фактів.



*Мал. 4. Сховище з архітектурою “Cніжинка”*

*Код форм та результати роботи програми наведено у додатку 3.*

**Висновки**

В даній курсовій роботі визначено та реалізовано базу даних для студентів проекту з використанням Microsoft SQL Server. Обрано реляційний підхід для моделювання структури даних, здійснено нормалізацію для підтримки цілісності та уникнення аномалій. Застосовані обмеження (constraint) для контролю цілісності даних, такі як унікальність, непорожність та перевірка умов.

Велика увага приділена розробці зручного графічного інтерфейсу користувача, який сприяє легкій взаємодії з системою. Використана мова запитів Transact-SQL для ефективного взаємодії з базою даних. Розроблена фізична модель бази даних відповідає вимогам Microsoft SQL Server, забезпечуючи оптимальні структури зберігання даних та методи доступу.

Загальна мета полягала в створенні ефективної, цілісної та безпечної бази даних для задоволення інформаційних потреб користувачів в галузі освітнього проекту. Отримана система дозволяє ефективно обробляти дані, забезпечуючи надійність та зручність використання.

**ЛІТЕРАТУРА**

1. Роберт, В. Програмування Баз Даних Microsoft SQL Server 2005 [Текст] / В. Роберт. - ООО.: "І.Д.Вільямс", 2007. – 832 с.
2. Іцик, Бєн-Ган. Microsoft SQL Server 2012. Основи T-SQL [Текст] / Б. Іцик. - [Пер. з англ. М. А. Райтман]. – Москва: Ексмо, 2015. – 400 с.
3. Андрусевич, І. С. Основи роботи з базами даних та SQL [Текст] / І. С. Андрусевич. - Видавництво "Нова книга", 2019. - 320 с.
4. Грофф, Майкл. SQL для простих смертных [Текст] / Майкл Грофф, Пол Вайнберг. - [Пер. з англ. А. Осовської] - Київ: Добра книга, 2017. - 416 с.
5. Фейлд, Кріс. Використання Microsoft SQL Server 2016. [Текст] / Кріс Фейлд, Шон Харінгтон. - [пер. з англ. А. П. Грінберга]. - СПб: Пітер, 2017. - 416 с.
6. Джонсон, Деніел. Програмування віконних додатків на C# для Microsoft Windows [Текст] / Деніел Джонсон. - [Пер. з англ. О. В. Моренець]. - Київ: Видавничий дім "Гельветика", 2018. - 464 с.
7. Краббе, Альфред. Програмування на C# для професійних програмістів [Текст] / Альфред Краббе. - [Пер. з англ. Д. С. Шерстюк]. - Київ: Видавництво "БІНОМ", 2019. - 560 с.

**ДОДАТКИ****Додаток 1**

Скрипт створення бази даних.

GO

USE master

GO

IF NOT EXISTS(SELECT \* FROM sys.databases WHERE NAME = 'StudentsDeploy')

BEGIN

CREATE DATABASE StudentsDeploy

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'StudentsDeploy',

FILENAME = N'C:\DB\StudentsDeployment.mdf' ,

SIZE = 8192KB ,

MAXSIZE = UNLIMITED,

FILEGROWTH = 15% ),

FILEGROUP StorageFileGroup

(

NAME = N'StudentsDeploySG',

FILENAME = N'C:\DB\StudentsDeploymentSG.ndf' ,

SIZE = 8192KB ,

MAXSIZE = UNLIMITED,

FILEGROWTH = 15% )

LOG ON

( NAME = N'StudentsDeploy\_log',

FILENAME = N'C:\DB\StudentsDeploymentLog.txt' ,

SIZE = 8192KB ,

MAXSIZE = 512MB ,

FILEGROWTH = 8192KB );

END

GO

USE StudentsDeploy

GO

IF (NOT EXISTS (SELECT \* FROM INFORMATION\_SCHEMA.TABLES WHERE TABLE\_SCHEMA = 'StudentsDeploySch'))

BEGIN EXEC sp\_executesql N'CREATE SCHEMA StudentsDeploySch' END

GO

IF OBJECT\_ID (N'StudentsDeploySch.Student`s stat', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.[Student`s stat]

GO

IF OBJECT\_ID (N'StudentsDeploySch.Students', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.Students

GO

IF OBJECT\_ID (N'StudentsDeploySch.Departments', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.Departments

GO

IF OBJECT\_ID (N'StudentsDeploySch.Subjects\_plan', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.Subjects\_plan

GO

IF OBJECT\_ID (N'StudentsDeploySch.TeachersSubjects', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.TeachersSubjects

GO

IF OBJECT\_ID (N'StudentsDeploySch.Teachers', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.Teachers

GO

IF OBJECT\_ID (N'StudentsDeploySch.Teacher\_ranks', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.Teacher\_ranks

GO

IF OBJECT\_ID (N'StudentsDeploySch.Disciplines', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.Disciplines

GO

CREATE TABLE StudentsDeploySch.Departments(

id int IDENTITY(1,1) NOT NULL UNIQUE,

Deanerys\_auditorium int NOT NULL,

Counter\_speciality int NOT NULL CHECK(Counter\_speciality>=0) DEFAULT(5),

Name nvarchar(50) NOT NULL DEFAULT('TK'),

CONSTRAINT PK\_Departments PRIMARY KEY CLUSTERED(id ASC) WITH (IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

CREATE TABLE StudentsDeploySch.Disciplines(

id int IDENTITY(1,1) NOT NULL UNIQUE,

Name nvarchar(30) NOT NULL,

CONSTRAINT PK\_Disciplines PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

CREATE TABLE StudentsDeploySch.Students(

id int IDENTITY(1,1) NOT NULL UNIQUE,

Age int NOT NULL CHECK(Age>=0),

FirstName nvarchar(50) NOT NULL,

SecondName nvarchar(50) NOT NULL,

department\_id int NOT NULL,

CONSTRAINT PK\_Students PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [PRIMARY],

CONSTRAINT [FK\_Students\_Departments] FOREIGN KEY([department\_id])

REFERENCES StudentsDeploySch.Departments ([id])

) ON [PRIMARY]

GO

ALTER TABLE [StudentsDeploySch].[Students] CHECK CONSTRAINT [FK\_Students\_Departments]

GO

CREATE TABLE StudentsDeploySch.Subjects\_plan(

id int IDENTITY(1,1) NOT NULL UNIQUE,

id\_discipline int NOT NULL,

Hours int NOT NULL CHECK(Hours>=0),

AuditoriumHours int NOT NULL CHECK(AuditoriumHours>=0),

CONSTRAINT PK\_Subjects PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [PRIMARY],

CONSTRAINT [FK\_Subjects\_plan\_Disciplines] FOREIGN KEY([id\_discipline])

REFERENCES StudentsDeploySch.Disciplines ([id])

) ON [PRIMARY]

GO

ALTER TABLE [StudentsDeploySch].[Subjects\_plan] CHECK CONSTRAINT [FK\_Subjects\_plan\_Disciplines]

GO

CREATE TABLE StudentsDeploySch.[Student`s stat](

id int IDENTITY(1,1) NOT NULL UNIQUE,

Semestr int NOT NULL CHECK(Semestr>=0),

Type int NOT NULL ,

Rating float NOT NULL,

students\_id int NOT NULL,

subject\_id int NOT NULL,

CONSTRAINT [PK\_Student`s stat] PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [PRIMARY],

CONSTRAINT CK\_Students\_type CHECK (Type='MK' OR Type='PK'),

CONSTRAINT [FK\_Student`s stat\_Students] FOREIGN KEY([students\_id])

REFERENCES StudentsDeploySch.Students ([id]),

CONSTRAINT [FK\_Student`s stat\_Subjects] FOREIGN KEY([subject\_id])

REFERENCES StudentsDeploySch.Subjects\_plan ([id])

)

GO

ALTER TABLE StudentsDeploySch.[Student`s stat] CHECK CONSTRAINT [FK\_Student`s stat\_Students]

GO

ALTER TABLE StudentsDeploySch.[Student`s stat] CHECK CONSTRAINT [FK\_Student`s stat\_Subjects]

GO

CREATE TABLE StudentsDeploySch.Teacher\_ranks(

id int IDENTITY(1,1) NOT NULL UNIQUE,

rank\_name nvarchar(50) NOT NULL,

CONSTRAINT PK\_Teacher\_ranks PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [StorageFileGroup]

) ON [StorageFileGroup]

GO

CREATE TABLE StudentsDeploySch.Teachers(

id int IDENTITY(1,1) NOT NULL UNIQUE,

FirstName nvarchar(30) NOT NULL,

SecondName nvarchar(30) NOT NULL,

Teacher\_rank\_id int NOT NULL,

CONSTRAINT PK\_Teachers PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [StorageFileGroup],

CONSTRAINT [FK\_Teachers\_Teacher\_ranks] FOREIGN KEY([Teacher\_rank\_id])

REFERENCES StudentsDeploySch.[Teacher\_ranks] ([id])

) ON [StorageFileGroup]

GO

ALTER TABLE StudentsDeploySch.[Teachers] CHECK CONSTRAINT [FK\_Teachers\_Teacher\_ranks]

GO

CREATE TABLE StudentsDeploySch.TeachersSubjects(

id int IDENTITY(1,1) NOT NULL UNIQUE,

idDiscipline int NOT NULL,

idTeachers int NOT NULL,

CONSTRAINT PK\_TeachersSubjects PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [StorageFileGroup],

CONSTRAINT [FK\_TeachersSubjects\_Disciplines] FOREIGN KEY([idDiscipline])

REFERENCES StudentsDeploySch.[Disciplines] ([id]),

CONSTRAINT [FK\_TeachersSubjects\_Teachers] FOREIGN KEY([idTeachers])

REFERENCES StudentsDeploySch.[Teachers] ([id])

) ON [StorageFileGroup]

GO

ALTER TABLE StudentsDeploySch.[TeachersSubjects] CHECK CONSTRAINT [FK\_TeachersSubjects\_Disciplines]

GO

ALTER TABLE StudentsDeploySch.[TeachersSubjects] CHECK CONSTRAINT [FK\_TeachersSubjects\_Teachers]

GO

IF OBJECT\_ID (N'StudentsDeploySch.TypeSubject', N'U') IS NOT NULL

DROP TABLE StudentsDeploySch.TypeSubject

GO

CREATE TABLE StudentsDeploySch.TypeSubject(

id int IDENTITY(1,1) NOT NULL UNIQUE,

Name nvarchar(30) NOT NULL,

CONSTRAINT PK\_TypeSubject PRIMARY KEY CLUSTERED

(

id ASC

)WITH (IGNORE\_DUP\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

INSERT INTO StudentsDeploySch.TypeSubject(Name) VALUES ('MK'), ('PK')

ALTER TABLE StudentsDeploySch.[Student`s stat]

ADD TypeI int NULL

GO

UPDATE StudentsDeploySch.[Student`s stat]

SET TypeI = CASE WHEN Type = 'MK' THEN 1 WHEN Type = 'PK' THEN 2 END

GO

ALTER TABLE StudentsDeploySch.[Student`s stat]

DROP CONSTRAINT CK\_Students\_type

GO

ALTER TABLE StudentsDeploySch.[Student`s stat]

DROP COLUMN Type

GO

EXEC sp\_rename 'StudentsDeploySch.[Student`s stat].TypeI', 'Type', 'COLUMN'

GO

ALTER TABLE StudentsDeploySch.[Student`s stat]

ALTER COLUMN Type int NOT NULL

GO

ALTER TABLE StudentsDeploySch.[Student`s stat]

ADD CONSTRAINT FK\_Type FOREIGN KEY(Type) REFERENCES StudentsDeploySch.TypeSubject(id)

GO

ALTER TABLE StudentsDeploySch.[Student`s stat] CHECK CONSTRAINT FK\_Type

GO

UPDATE StudentsDeploySch.Subjects\_plan

SET AuditoriumHours = Hours/2 WHERE id\_discipline%2 = 1

DELETE

FROM StudentsDeploySch.[Student`s stat]

WHERE students\_id = 2

DELETE

FROM StudentsDeploySch.Students

WHERE id = 2

INSERT INTO StudentsDeploySch.Departments(Name,Counter\_speciality, Deanerys\_auditorium)

VALUES ('TK', 4, 435), ('UPP',1,420), ('Mechaniks',3,246), ('PGS',10,315), ('Teach',4,5413), ('MT', 3,123), ('Lokomotives',2,299),

('PM', 1, 518), ('GPM',3,216), ('Economics', 2,222)

GO

INSERT INTO StudentsDeploySch.Disciplines(Name)

VALUES ('Algebra'),('Geometria'),('Phiziks'),('Informatica'),('Programming'), ('Buildings'),('Electricity'),('Databases'),('Linear algebra'),

('OOP'),('Economics')

GO

INSERT INTO StudentsDeploySch.Teacher\_ranks(rank\_name)

VALUES ('Aspirant'), ('Teacher'), ('Senior Teacher'), ('Docent'), ('Proffesor')

GO

INSERT INTO StudentsDeploySch.Teachers(SecondName,FirstName,Teacher\_rank\_id)

VALUES ('Ivanov','Ivan',5), ('Dmitriev','Vadim',4), ('Sergeev','Nikolay',4),('Alexandrova','Anna',2),('Parksenov','Fedor',3),

('Maksimova','Irina',1),('Nikolaenko', 'Pavel',2),('Andreev','Alexey',1), ('Viy','Alexandr',3),('Danilov','Sergey',5)

GO

INSERT INTO StudentsDeploySch.TeachersSubjects(idTeachers,idDiscipline)

VALUES (1,1),(1,2),(1,3),(2,2),(3,3),(5,4),(5,4),(6,10),(7,6),(8,8),(10,5),(9,10),(9,1)

GO

INSERT INTO StudentsDeploySch.Subjects\_plan(id\_discipline,Hours,AuditoriumHours)

VALUES (1,100,20),(2,200,100),(1,50,5),(3,300,50),(4,55,20), (6,10,5),(5,200,200),(6,300,100),(7,88,55),(8,100,90),(9,100,50),(10,10,5)

GO

INSERT INTO StudentsDeploySch.Students(SecondName,FirstName,Age,department\_id)

VALUES ('Anatolev','Nikolay',19,2),('Mihaylov','Danil',21,3),('Anatoleva','Elena',18,1),('Sidorov','Sergey',20,4),

('Amak','Maxim',20,3),('Topolev','Igor',19,1),('Saxev','Ivan',20,2),('Grigorev','Alexey',19,4),('Pirsova','Alexandra',21,3),

('Kipish','Anna',21,4)GO

INSERT INTO StudentsDeploySch.[Student`s stat](Semestr,Type,Rating,students\_id,subject\_id)

VALUES (1,1,78,6,1),(1,1,80,6,2),(1,1,68,6,3),(2,2,74,6,4),(1,1,69,6,5),(2,2,87,6,6),

(1,1,79,6,7),(2,2,80,6,8),(1,1,75,6,9),(2,2,78,6,10),(1,1,85,6,11),(2,2,70,6,12),

(1,1,82,6,13),(2,2,73,6,15)

GO

**Додаток 2**

Процедури для роботи з базою даних що використовуються у програмі.

/\*Процедура для добавления студентов\*/

IF OBJECT\_ID('InsertStudent', 'P') IS NOT NULL

DROP PROCEDURE InsertStudent;

GO

CREATE PROCEDURE InsertStudent

@age AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50),

@departmentid AS int,

@specialityid AS int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StudentsDeploySch.Students(Age, FirstName, SecondName, department\_id, speciality\_id)

VALUES(@age, @FirstName, @SecondName, @departmentid, @specialityid)

END;

GO

IF OBJECT\_ID('SelectStudent', 'P') IS NOT NULL

DROP PROCEDURE SelectStudent;

GO

CREATE PROCEDURE SelectStudent

AS

BEGIN

SET NOCOUNT ON;

SELECT st.id, st.FirstName AS Имя, st.SecondName AS Фаммилия, st.Age AS Возраст, dp.Name AS Факультет, sp.Name AS Специальность

FROM StudentsDeploySch.Students AS st

JOIN StudentsDeploySch.Departments AS dp ON st.department\_id = dp.id

JOIN StudentsDeploySch.Speciality AS sp ON st.speciality\_id = sp.id

END;

GO

IF OBJECT\_ID('SearchStudentByName', 'P') IS NOT NULL

DROP PROCEDURE SearchStudentByName;

GO

CREATE PROCEDURE SearchStudentByName

@SecondName AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT st.id AS ID, st.FirstName AS Имя, st.SecondName AS Фаммилия, st.Age AS Возраст, dp.Name AS Факультет, sp.Name AS Специальность

FROM StudentsDeploySch.Students AS st

JOIN StudentsDeploySch.Departments AS dp ON st.department\_id = dp.id

JOIN StudentsDeploySch.Speciality AS sp ON st.speciality\_id = sp.id

WHERE st.SecondName LIKE @SecondName+'%'

END;

GO

/\*Процедура для обновления студентов\*/

IF OBJECT\_ID('UpdateStudent', 'P') IS NOT NULL

DROP PROCEDURE UpdateStudent;

GO

CREATE PROCEDURE UpdateStudent

@id AS int,

@age AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50),

@departmentid AS int,

@specialityid AS int

AS

BEGIN

UPDATE StudentsDeploySch.Students

SET

age = @age,

FirstName = @FirstName,

SecondName = @SecondName,

department\_id = @departmentid,

speciality\_id = @specialityid

WHERE id = @id

END;

GO

/\*Процедура для удаления студентов\*/

IF OBJECT\_ID('DeleteStudent', 'P') IS NOT NULL

DROP PROCEDURE DeleteStudent;

GO

CREATE PROCEDURE DeleteStudent

@id AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50)

AS

BEGIN

DELETE StudentsDeploySch.Students

WHERE id = @id AND FirstName = @FirstName AND SecondName = @SecondName

END;

GO

/\*Процедура для проверки существования студента\*/

IF OBJECT\_ID('ExistStudent', 'P') IS NOT NULL

DROP PROCEDURE ExistStudent;

GO

CREATE PROCEDURE ExistStudent

@id AS int

AS

BEGIN

SELECT COUNT(st.id)

FROM StudentsDeploySch.Students AS st

WHERE st.id = @id

END;

GO

/\*Процедура для добавления преподавателей\*/

IF OBJECT\_ID('InsertTeachers', 'P') IS NOT NULL

DROP PROCEDURE InsertTeachers;

GO

CREATE PROCEDURE InsertTeachers

@FirstName AS nvarchar(30),

@SecondName AS nvarchar(30),

@Teachers\_runk\_id AS int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StudentsDeploySch.Teachers(FirstName, SecondName, Teacher\_rank\_id)

VALUES(@FirstName, @SecondName, @Teachers\_runk\_id)

END;

GO

/\*Процедура для вывода всех преподавателей в GridView\*/

IF OBJECT\_ID('SelectTeachers', 'P') IS NOT NULL

DROP PROCEDURE SelectTeachers;

GO

CREATE PROCEDURE SelectTeachers

AS

BEGIN

SET NOCOUNT ON;

SELECT teach.id, teach.FirstName AS Имя, teach.SecondName AS Фаммилия, runk.rank\_name AS Степень

FROM StudentsDeploySch.Teachers AS teach

JOIN StudentsDeploySch.Teacher\_ranks AS runk ON Teacher\_rank\_id = runk.id

END;

GO

/\*Процедура для поиска преподавателей по фамилии\*/

IF OBJECT\_ID('SearchTeachersByName', 'P') IS NOT NULL

DROP PROCEDURE SearchTeachersByName;

GO

CREATE PROCEDURE SearchTeachersByName

@SecondName AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT teach.id, teach.FirstName AS Имя, teach.SecondName AS Фаммилия, runk.rank\_name AS Степень

FROM StudentsDeploySch.Teachers AS teach

JOIN StudentsDeploySch.Teacher\_ranks AS runk ON Teacher\_rank\_id = runk.id

WHERE teach.SecondName LIKE @SecondName+'%'

END;

GO

/\*Процедура обновления данных о преподавателях\*/

IF OBJECT\_ID('UpdateTeachers', 'P') IS NOT NULL

DROP PROCEDURE UpdateTeachers;

GO

CREATE PROCEDURE UpdateTeachers

@id AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50),

@Teachers\_runk\_id AS int

AS

BEGIN

UPDATE StudentsDeploySch.Teachers

SET

FirstName = @FirstName,

SecondName = @SecondName,

Teacher\_rank\_id = @Teachers\_runk\_id

WHERE id = @id

END;

GO

/\*Процедура удаления данных о преподавателях\*/

IF OBJECT\_ID('DeleteTeachers', 'P') IS NOT NULL

DROP PROCEDURE DeleteTeachers;

GO

CREATE PROCEDURE DeleteTeachers

@id AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50)

AS

BEGIN

DELETE StudentsDeploySch.Teachers

WHERE id = @id AND FirstName = @FirstName AND SecondName = @SecondName

END;

GO

/\*Проверка на существование преподавателя\*/

IF OBJECT\_ID('ExistTeachers', 'P') IS NOT NULL

DROP PROCEDURE ExistTeachers;

GO

CREATE PROCEDURE ExistTeachers

@Teachers\_id AS int

AS

BEGIN

SELECT COUNT(id)

FROM StudentsDeploySch.Teachers

WHERE id = @Teachers\_id

END;

GO

IF OBJECT\_ID('InsertDisciplines', 'P') IS NOT NULL

DROP PROCEDURE InsertDisciplines;

GO

CREATE PROCEDURE InsertDisciplines

@Name AS nvarchar(30),

@Subject\_code AS smallint

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StudentsDeploySch.Disciplines(Name, Subject\_Code)

VALUES(@Name, @Subject\_code)

END;

GO

/\*Процедура для вывода всех дисциплин в GridView\*/

IF OBJECT\_ID('SelectDiscipline', 'P') IS NOT NULL

DROP PROCEDURE SelectDiscipline;

GO

CREATE PROCEDURE SelectDiscipline

AS

BEGIN

SET NOCOUNT ON;

SELECT dis.id, dis.Name, dis.Subject\_Code

FROM StudentsDeploySch.Disciplines AS dis

END;

GO

/\*Процедура для поиска дисциплин\*/

IF OBJECT\_ID('SearchDiscipline', 'P') IS NOT NULL

DROP PROCEDURE SearchDiscipline;

GO

CREATE PROCEDURE SearchDiscipline

@Name AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT dis.id, dis.Name, dis.Subject\_Code

FROM StudentsDeploySch.Disciplines AS dis

WHERE dis.Name LIKE @Name+'%'

END;

GO

/\*Процедура обновления дисциплин\*/

IF OBJECT\_ID('UpdateDiscipline', 'P') IS NOT NULL

DROP PROCEDURE UpdateDiscipline;

GO

CREATE PROCEDURE UpdateDiscipline

@id AS int,

@Name AS nvarchar(30),

@Subject\_code AS smallint

AS

BEGIN

UPDATE StudentsDeploySch.Disciplines

SET

Name = @Name,

Subject\_Code = @Subject\_code

WHERE id = @id

END;

GO

/\*Процедура удаления дисциплин\*/

IF OBJECT\_ID('DeleteDiscipline', 'P') IS NOT NULL

DROP PROCEDURE DeleteDiscipline;

GO

CREATE PROCEDURE DeleteDiscipline

@id AS int,

@Name AS nvarchar(80)

AS

BEGIN

DELETE StudentsDeploySch.Disciplines

WHERE id = @id AND Name = @Name

END;

GO

/\*----------СТАТИСТИКА---------------\*/

/\*Процедура для добавления оценок\*/

IF OBJECT\_ID('InsertStut', 'P') IS NOT NULL

DROP PROCEDURE InsertStut;

GO

CREATE PROCEDURE InsertStut

@Students\_id AS int,

@Subject\_id as int,

@Type as int,

@Semestr as int,

@Rating as float

AS

BEGIN

INSERT INTO StudentsDeploySch.[Student`s stat](Semestr, Type, Rating, students\_id, subject\_id)

VALUES(@Semestr, @Type, @Rating, @Students\_id, @Subject\_id)

END;

GO

/\*Процедура для добавления учебного плана для выбранного студента\*/

IF OBJECT\_ID('InsertStudentsPlan', 'P') IS NOT NULL

DROP PROCEDURE InsertStudentsPlan;

GO

CREATE PROCEDURE InsertStudentsPlan

@StudId AS int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO StudentsDeploySch.[Student`s stat](Semestr, Type, Rating, students\_id, subject\_id)

VALUES (1,1,0,@StudId,1),(2,2,0,@StudId,2),(1,1,0,@StudId,3),(2,2,0,@StudId,4),(1,1,0,@StudId,5),

(2,2,0,@StudId,6),(1,1,0,@StudId,7),(2,2,0,@StudId,8),(1,1,0,@StudId,9),(2,2,0,@StudId,10),

(1,1,0,@StudId,11),(2,2,0,@StudId,12),(1,1,0,@StudId,13),(2,2,0,@StudId,15)

END;

GO

/\*Процедура для вывода всей статистики всех студентов в GridView\*/

IF OBJECT\_ID('SelectALLStudentStut', 'P') IS NOT NULL

DROP PROCEDURE SelectALLStudentStut;

GO

CREATE PROCEDURE SelectALLStudentStut

AS

BEGIN

SET NOCOUNT ON;

SELECT Stut.id AS ID, St.FirstName AS Имя, St.SecondName AS Фамилия, Dep.Name AS Факультет, Sp.Name AS Специальность,

Dis.Name AS Дисциплина, stut.Semestr AS Семестр, Ty.Name AS Тип, Stut.Rating AS Рейтинг, St.id AS StudentsId

FROM StudentsDeploySch.[Student`s stat] AS Stut

JOIN StudentsDeploySch.Students AS St ON stut.students\_id = St.id

JOIN StudentsDeploySch.Disciplines AS Dis ON stut.subject\_id = Dis.id

JOIN StudentsDeploySch.TypeSubject AS Ty ON Stut.Type = Ty.id

JOIN StudentsDeploySch.Departments AS Dep ON St.department\_id = Dep.id

JOIN StudentsDeploySch.Speciality AS Sp ON St.speciality\_id = Sp.id

END;

GO

IF OBJECT\_ID('SelectStudentStut', 'P') IS NOT NULL

DROP PROCEDURE SelectStudentStut;

GO

CREATE PROCEDURE SelectStudentStut

@id AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT Stut.id AS ID, St.FirstName AS Имя, St.SecondName AS Фамилия, Dep.Name AS Факультет, Sp.Name AS Специальность,

Dis.Name AS Дисциплина, stut.Semestr AS Семестр, Ty.Name AS Тип, Stut.Rating AS Рейтинг, St.id AS StudentsId

FROM StudentsDeploySch.Students AS St

LEFT JOIN StudentsDeploySch.[Student`s stat] AS Stut ON stut.students\_id = St.id

JOIN StudentsDeploySch.Disciplines AS Dis ON stut.subject\_id = Dis.id

JOIN StudentsDeploySch.TypeSubject AS Ty ON Stut.Type = Ty.id

JOIN StudentsDeploySch.Departments AS Dep ON St.department\_id = Dep.id

JOIN StudentsDeploySch.Speciality AS Sp ON St.speciality\_id = Sp.id

WHERE stut.students\_id = @id AND St.FirstName = @FirstName AND St.SecondName = @SecondName

END;

GO

/\*Процедура для вывода задолженностей выбранного студента в GridView\*/

IF OBJECT\_ID('SelectStudentArrears', 'P') IS NOT NULL

DROP PROCEDURE SelectStudentArrears;

GO

CREATE PROCEDURE SelectStudentArrears

@id AS int,

@FirstName AS nvarchar(50),

@SecondName AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT Stut.id AS ID, St.FirstName AS Имя, St.SecondName AS Фамилия, Dep.Name AS Факультет, Sp.Name AS Специальность,

Dis.Name AS Дисциплина, stut.Semestr AS Семестр, Ty.Name AS Тип, Stut.Rating AS Рейтинг, St.id AS StudentsId

FROM StudentsDeploySch.Students AS St

JOIN StudentsDeploySch.[Student`s stat] AS Stut ON stut.students\_id = St.id

JOIN StudentsDeploySch.Disciplines AS Dis ON stut.subject\_id = Dis.id

JOIN StudentsDeploySch.TypeSubject AS Ty ON Stut.Type = Ty.id

JOIN StudentsDeploySch.Departments AS Dep ON St.department\_id = Dep.id

JOIN StudentsDeploySch.Speciality AS Sp ON St.speciality\_id = Sp.id

WHERE stut.students\_id = @id AND St.FirstName = @FirstName AND St.SecondName = @SecondName AND stut.Rating = 0

END;

GO

/\*Процедура для вывода задолженностей всех студентов в GridView\*/

IF OBJECT\_ID('SelectALLStudentArrears', 'P') IS NOT NULL

DROP PROCEDURE SelectALLStudentArrears;

GO

CREATE PROCEDURE SelectALLStudentArrears

AS

BEGIN

SET NOCOUNT ON;

SELECT Stut.id AS ID, St.FirstName AS Имя, St.SecondName AS Фамилия, Dep.Name AS Факультет, Sp.Name AS Специальность,

Dis.Name AS Дисциплина, stut.Semestr AS Семестр, Ty.Name AS Тип, Stut.Rating AS Рейтинг, St.id AS StudentsId

FROM StudentsDeploySch.[Student`s stat] AS Stut

JOIN StudentsDeploySch.Students AS St ON stut.students\_id = St.id

JOIN StudentsDeploySch.Disciplines AS Dis ON stut.subject\_id = Dis.id

JOIN StudentsDeploySch.TypeSubject AS Ty ON Stut.Type = Ty.id

JOIN StudentsDeploySch.Departments AS Dep ON St.department\_id = Dep.id

JOIN StudentsDeploySch.Speciality AS Sp ON St.speciality\_id = Sp.id AND stut.Rating = 0

END;

GO

/\*Процедура обновления успеваемости\*/

IF OBJECT\_ID('UpdateStut', 'P') IS NOT NULL

DROP PROCEDURE UpdateStut;

GO

CREATE PROCEDURE UpdateStut

@id AS int,

@Students\_id AS int,

@Subject\_id as int,

@Type as int,

@Semestr as int,

@Rating as float

AS

BEGIN

SET NOCOUNT ON;

UPDATE StudentsDeploySch.[Student`s stat]

SET

Semestr = @Semestr,

Type = @Type,

Rating = @Rating,

students\_id = @Students\_id,

subject\_id = @Subject\_id

WHERE id = @id

END;

GO

IF OBJECT\_ID('ExistStudentStut', 'P') IS NOT NULL

DROP PROCEDURE ExistStudentStut;

GO

CREATE PROCEDURE ExistStudentStut

@id AS int

AS

BEGIN

SELECT COUNT(stut.id)

FROM StudentsDeploySch.[Student`s stat] AS stut

WHERE stut.students\_id = @id

END;

GO

/\*Процедура удаления учебного плана студента\*/

IF OBJECT\_ID('DeleteStudentsPlan', 'P') IS NOT NULL

DROP PROCEDURE DeleteStudentsPlan;

GO

CREATE PROCEDURE DeleteStudentsPlan

@student\_id AS int

AS

BEGIN

SET NOCOUNT ON;

DELETE StudentsDeploySch.[Student`s stat]

WHERE students\_id = @student\_id

END;

GO

/\*Процедура удаления оценки студента\*/

IF OBJECT\_ID('DeleteStudentsMark', 'P') IS NOT NULL

DROP PROCEDURE DeleteStudentsMark;

GO

CREATE PROCEDURE DeleteStudentsMark

@id AS int

AS

BEGIN

DELETE StudentsDeploySch.[Student`s stat]

WHERE id = @id

END;

GO

/\*-----------ПЛАН ДИСЦИПЛИН---------\*/

/\*Добавление плана\*/

IF OBJECT\_ID('InsertSubPlan', 'P') IS NOT NULL

DROP PROCEDURE InsertSubPlan;

GO

CREATE PROCEDURE InsertSubPlan

@Discipline\_id AS int,

@Hours as int,

@AuditHours as int

AS

BEGIN

INSERT INTO StudentsDeploySch.Subjects\_plan(id\_discipline, Hours, AuditoriumHours)

VALUES(@Discipline\_id, @Hours, @AuditHours)

END;

GO

/\*Проверка на существование дисциплины\*/

IF OBJECT\_ID('ExistDiscipline', 'P') IS NOT NULL

DROP PROCEDURE ExistDiscipline;

GO

CREATE PROCEDURE ExistDiscipline

@Discipline\_id AS int

AS

BEGIN

SELECT COUNT(dis.id)

FROM StudentsDeploySch.Disciplines AS dis

WHERE dis.id = @Discipline\_id

END;

GO

IF OBJECT\_ID('SearchDisciplinePlan', 'P') IS NOT NULL

DROP PROCEDURE SearchDisciplinePlan;

GO

CREATE PROCEDURE SearchDisciplinePlan

@Name AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT splan.id, dis.Name AS Дисциплина, dis.Subject\_Code AS Код, splan.Hours AS Часы, splan.AuditoriumHours AS [Аудиторные часы]

FROM StudentsDeploySch.Disciplines AS dis

JOIN StudentsDeploySch.Subjects\_plan AS splan ON dis.id = splan.id\_discipline

WHERE dis.Name LIKE @Name+'%'

END;

GO

IF OBJECT\_ID('SelectDisciplinePlan', 'P') IS NOT NULL

DROP PROCEDURE SelectDisciplinePlan;

GO

CREATE PROCEDURE SelectDisciplinePlan

AS

BEGIN

SELECT splan.id, dis.Name AS Дисциплина, dis.Subject\_Code AS Код, splan.Hours AS Часы, splan.AuditoriumHours AS [Аудиторные часы]

FROM StudentsDeploySch.Disciplines AS dis

JOIN StudentsDeploySch.Subjects\_plan AS splan ON dis.id = splan.id\_discipline

END;

GO

/\*Процедура для обновления плана дисциплин\*/

IF OBJECT\_ID('UpdateDisciplinePlan', 'P') IS NOT NULL

DROP PROCEDURE UpdateDisciplinePlan;

GO

CREATE PROCEDURE UpdateDisciplinePlan

@Plan\_id AS int,

@Hours as int,

@AuditHours as int

AS

BEGIN

SET NOCOUNT ON;

UPDATE StudentsDeploySch.Subjects\_plan

SET

Hours = @Hours,

AuditoriumHours = @AuditHours

WHERE id = @Plan\_id

END;

GO

/\*Процедура для удаления плана дисциплин\*/

IF OBJECT\_ID('DeleteSubjectPlan', 'P') IS NOT NULL

DROP PROCEDURE DeleteSubjectPlan;

GO

CREATE PROCEDURE DeleteSubjectPlan

@id AS int

AS

BEGIN

DELETE StudentsDeploySch.Subjects\_plan

WHERE id = @id

END;

GO

/\*-----ФАКУЛЬТЕТЫ И СПЕЦИАЛЬНОСТИ-----\*/

/\*Добавление факультета\*/

IF OBJECT\_ID('InsertDepartment', 'P') IS NOT NULL

DROP PROCEDURE InsertDepartment;

GO

CREATE PROCEDURE InsertDepartment

@Auditorium AS int,

@Name AS nvarchar(50)

AS

BEGIN

INSERT INTO StudentsDeploySch.Departments(Deanerys\_auditorium, Name)

VALUES(@Auditorium, @Name)

END;

GO

/\*Процедура для выборки факультетов\*/

IF OBJECT\_ID('SelectDepartments', 'P') IS NOT NULL

DROP PROCEDURE SelectDepartments;

GO

CREATE PROCEDURE SelectDepartments

AS

BEGIN

SELECT dep.id, dep.Name AS Факультет, dep.Deanerys\_auditorium AS Аудитория

FROM StudentsDeploySch.Departments AS dep

END;

GO

/\*Процедура для поиска факультета\*/

IF OBJECT\_ID('SearchDepartments', 'P') IS NOT NULL

DROP PROCEDURE SearchDepartments;

GO

CREATE PROCEDURE SearchDepartments

@Name AS nvarchar(50)

AS

BEGIN

SET NOCOUNT ON;

SELECT dep.id, dep.Name AS Факультет, dep.Deanerys\_auditorium AS Аудитория

FROM StudentsDeploySch.Departments AS dep

WHERE dep.Name LIKE @Name+'%'

END;

GO

IF OBJECT\_ID('UpdateDeparments', 'P') IS NOT NULL

DROP PROCEDURE UpdateDeparments;

GO

CREATE PROCEDURE UpdateDeparments

@id AS int,

@Name as nvarchar(50),

@Auditorium as int

AS

BEGIN

SET NOCOUNT ON;

UPDATE StudentsDeploySch.Departments

SET

Name = @Name,

Deanerys\_auditorium = @Auditorium

WHERE id = @id

END;

GO

/\*Проверка на существование факультета\*/

IF OBJECT\_ID('ExistDepartment', 'P') IS NOT NULL

DROP PROCEDURE ExistDepartment;

GO

CREATE PROCEDURE ExistDepartment

@Department\_id AS int

AS

BEGIN

SELECT COUNT(id)

FROM StudentsDeploySch.Departments

WHERE id = @Department\_id

END;

GO

/\*Процедура удаления факультетов\*/

IF OBJECT\_ID('DeleteDepartments', 'P') IS NOT NULL

DROP PROCEDURE DeleteDepartments;

GO

CREATE PROCEDURE DeleteDepartments

@id AS int

AS

BEGIN

DELETE StudentsDeploySch.Departments

WHERE id = @id

END;

GO

/\*Добавление специальности\*/

IF OBJECT\_ID('InsertSpeciality', 'P') IS NOT NULL

DROP PROCEDURE InsertSpeciality;

GO

CREATE PROCEDURE InsertSpeciality

@Id\_department AS int,

@Name AS nvarchar(100)

AS

BEGIN

INSERT INTO StudentsDeploySch.Speciality(IdDepartment, Name)

VALUES(@Id\_department, @Name)

END;

GO

/\*Процедура для выборки специальностей\*/

IF OBJECT\_ID('SelectSpeciality', 'P') IS NOT NULL

DROP PROCEDURE SelectSpeciality;

GO

CREATE PROCEDURE SelectSpeciality

AS

BEGIN

SELECT sp.id, dep.Name AS Факультет, sp.Name AS Специальность, sp.IdDepartment, dep.Deanerys\_auditorium

FROM StudentsDeploySch.Speciality AS sp

JOIN StudentsDeploySch.Departments AS dep ON sp.IdDepartment = dep.id

END;

GO

/\*Процедура для поиска специальностей\*/

IF OBJECT\_ID('SearchSpeciality', 'P') IS NOT NULL

DROP PROCEDURE SearchSpeciality;

GO

CREATE PROCEDURE SearchSpeciality

@Name AS nvarchar(100)

AS

BEGIN

SET NOCOUNT ON;

SELECT sp.id, dep.Name AS Факультет, sp.Name AS Специальность, sp.IdDepartment, dep.Deanerys\_auditorium

FROM StudentsDeploySch.Speciality AS sp

JOIN StudentsDeploySch.Departments AS dep ON sp.IdDepartment = dep.id

WHERE sp.Name LIKE @Name+'%'

END;

GO

/\*Процедура обновления специальностей\*/

IF OBJECT\_ID('UpdateSpeciality', 'P') IS NOT NULL

DROP PROCEDURE UpdateSpeciality;

GO

CREATE PROCEDURE UpdateSpeciality

@id AS int,

@Id\_department AS int,

@Name AS nvarchar(100)

AS

BEGIN

SET NOCOUNT ON;

UPDATE StudentsDeploySch.Speciality

SET

IdDepartment = @Id\_department,

Name = @Name

WHERE id = @id

END;

GO

IF OBJECT\_ID('ExistSpeciality', 'P') IS NOT NULL

DROP PROCEDURE ExistSpeciality;

GO

CREATE PROCEDURE ExistSpeciality

@Speciality\_id AS int

AS

BEGIN

SELECT COUNT(id)

FROM StudentsDeploySch.Speciality

WHERE id = @Speciality\_id

END;

GO

/\*Процедура для удаления специальности\*/

IF OBJECT\_ID('DeleteSpeciality', 'P') IS NOT NULL

DROP PROCEDURE DeleteSpeciality;

GO

CREATE PROCEDURE DeleteSpeciality

@id AS int

AS

BEGIN

DELETE StudentsDeploySch.Speciality

WHERE id = @id

END;

GO

/\*Процедура для назначения дисциплин преподавателям\*/

IF OBJECT\_ID('InsertDisTeach', 'P') IS NOT NULL

DROP PROCEDURE InsertDisTeach;

GO

CREATE PROCEDURE InsertDisTeach

@TeachId AS int,

@DisId AS nvarchar(100)

AS

BEGIN

INSERT INTO StudentsDeploySch.TeachersSubjects(idTeachers, idDiscipline)

VALUES(@TeachId, @DisId)

END;

GO

/\*Процедура для вывода дисциплин, которые ведет выбранный преподаватель\*/

IF OBJECT\_ID('SelectTeachDis', 'P') IS NOT NULL

DROP PROCEDURE SelectTeachDis;

GO

CREATE PROCEDURE SelectTeachDis

@TeachId AS int

AS

BEGIN

SELECT tsub.id, tich.FirstName AS Имя, tich.SecondName AS Фамилия, dis.Name AS Дисциплина, tich.id

FROM StudentsDeploySch.TeachersSubjects AS tsub

JOIN StudentsDeploySch.Teachers AS tich ON tsub.idTeachers = tich.id

JOIN StudentsDeploySch.Disciplines AS dis ON tsub.idDiscipline = dis.id

WHERE tsub.idTeachers = @TeachId

END;

GO

/\*процедура для вывборки дисциплин которые ведут все преподаватели\*/

IF OBJECT\_ID('SelectAllTeachDis', 'P') IS NOT NULL

DROP PROCEDURE SelectAllTeachDis;

GO

CREATE PROCEDURE SelectAllTeachDis

AS

BEGIN

SELECT tsub.id, tich.FirstName AS Имя, tich.SecondName AS Фамилия, dis.Name AS Дисциплина, tich.id

FROM StudentsDeploySch.TeachersSubjects AS tsub

JOIN StudentsDeploySch.Teachers AS tich ON tsub.idTeachers = tich.id

JOIN StudentsDeploySch.Disciplines AS dis ON tsub.idDiscipline = dis.id

END;

GO

/\*Процедура для обновления преподавательских дисциплин\*/

IF OBJECT\_ID('UpdateDisTeach', 'P') IS NOT NULL

DROP PROCEDURE UpdateDisTeach;

GO

CREATE PROCEDURE UpdateDisTeach

@id AS int,

@Id\_Teacher AS int,

@Id\_discipline AS int

AS

BEGIN

SET NOCOUNT ON;

UPDATE StudentsDeploySch.TeachersSubjects

SET

idTeachers = @Id\_Teacher,

idDiscipline = @Id\_discipline

WHERE id = @id

END;

GO

/\*Процедура для удаления дисциплины у преподавателя\*/

IF OBJECT\_ID('DeleteDisTeach', 'P') IS NOT NULL

DROP PROCEDURE DeleteDisTeach;

GO

CREATE PROCEDURE DeleteDisTeach

@id AS int

AS

BEGIN

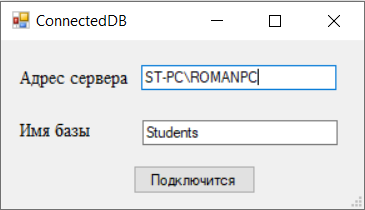
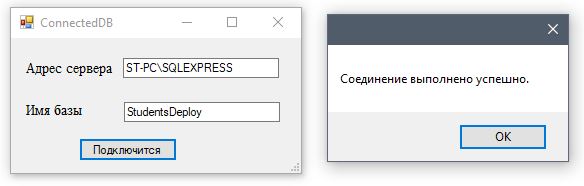
DELETE StudentsDeploySch.TeachersSubjects

WHERE id = @id

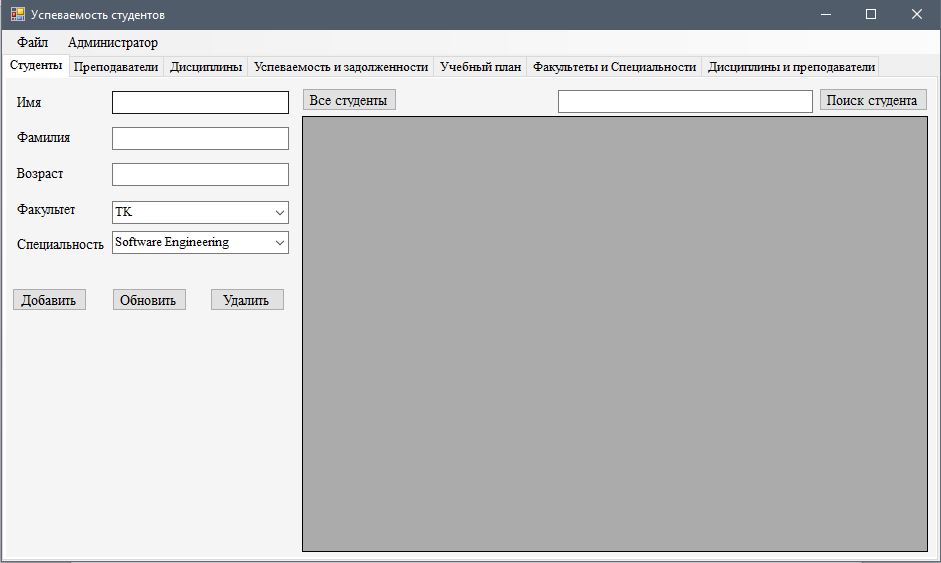
END;

GO

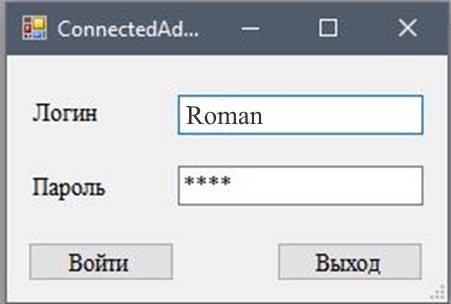
**Додаток 3**

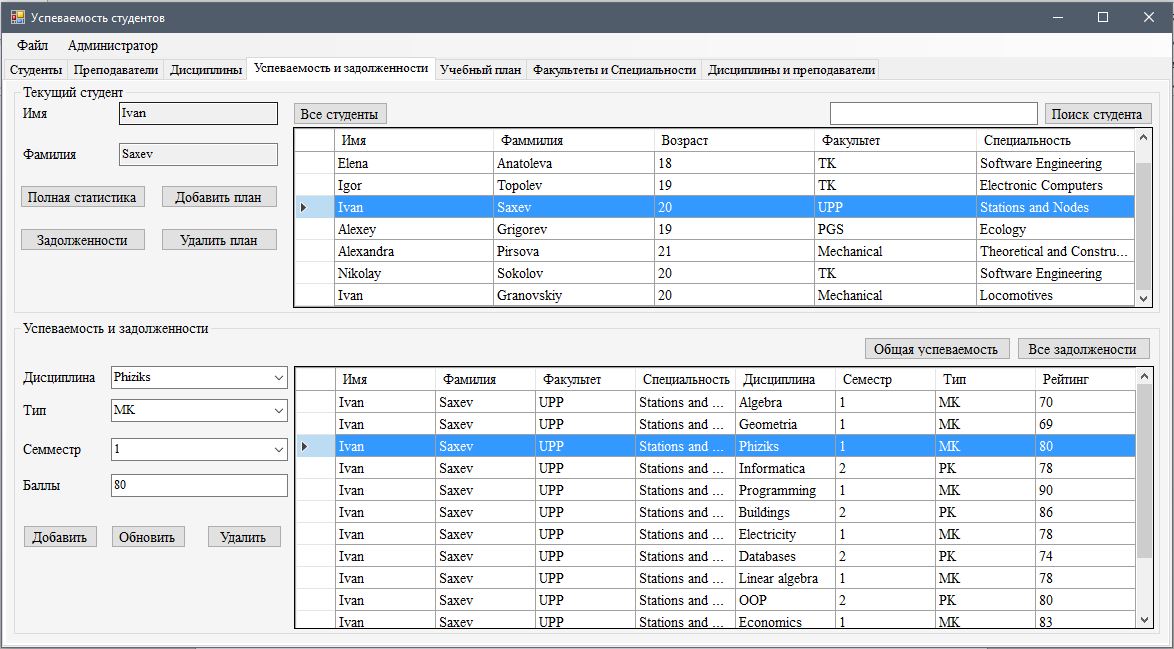
Мал. 5. Підключення до бази даних.



Мал. 6. Головне вікно програми.



Мал. 7. Вікно авторизації адміністратора.



Мал. 8. Перегляд успішності студента.

***Запити до схеми «Зірка» з результатами виконання.***

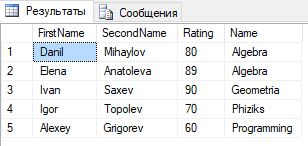
SELECT stud.FirstName, stud.SecondName, inf.Rating, dis.Name

FROM dbo.Info\_Facts AS inf

JOIN dbo.Disciplines\_Met AS dis ON inf.ID\_Disciplines = dis.id

JOIN dbo.Students\_Met AS stud ON stud.id = inf.ID\_Students

GO



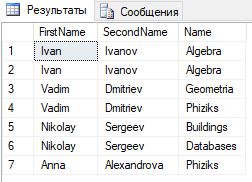
SELECT tich.FirstName, tich.SecondName, dis.Name

FROM dbo.Info\_Facts AS inf

JOIN dbo.Teachers\_Met AS tich ON tich.id = inf.ID\_Teachers

JOIN dbo.Disciplines\_Met AS dis ON dis.id = inf.ID\_Disciplines

GO



***Запити до схеми «Сніжинка» з результатами виконання***

SELECT st.FirstName, st.SecondName, dep.Name, sp.Name

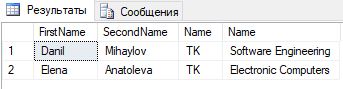
FROM dbo.Info\_Facts AS inf

JOIN dbo.Students\_Met AS st ON st.id = inf.ID\_Students

JOIN dbo.Departments\_Met AS dep ON dep.id = inf.ID\_Departments

JOIN dbo.Speciality\_Met AS sp ON sp.id = inf.ID\_Speciality

WHERE dep.Name = 'TK'



SELECT st.FirstName, st.SecondName, dis.Name, inf.Rating, inf.Type

FROM dbo.Students\_Met AS st

JOIN dbo.Info\_Facts AS inf ON inf.ID\_Students = st.id

JOIN dbo.Disciplines\_Met AS dis ON inf.ID\_Disciplines = dis.id

WHERE inf.Rating = 0 AND inf.Type = 'PK'

