Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

Курсова робота

з курсу «Надвеликі бази даних»

за темою «Розробка та аналіз надвеликої бази даних з використанням технологій Microsoft SQL Server»

Виконав:

студентка групи ПМ-42

Леонід ЧІСТЯКОВ

Перевірив:

Богдан ЛЮБІНСЬКІЙ

_____

(дата) (підпис викладача)

Львів – 2026 р.

Анотація

Курсова робота присвячена розробці та аналізу надвеликої бази даних для предметної області «Студент (Деканат)» з використанням технологій Microsoft SQL Server. У роботі реалізовано повний цикл створення інформаційної системи, починаючи з аналізу предметної області та проектування реляційної бази даних, і завершуючи побудовою сховища даних, OLAP-куба та аналітичних звітів.

У межах курсової роботи спроєктовано базу даних, що охоплює основні сутності деканату: студентів, групи, викладачів, предмети, оцінки, хобі, книги та проживання у гуртожитку. База даних наповнена великим обсягом реалістичних даних (понад 10 000 студентів і 500 000 записів оцінок) з урахуванням часової історії навчання за п’ять років.

Для підготовки аналітичних даних розроблено ETL-процеси за допомогою SQL Server Integration Services, виконано очищення, трансформацію та агрегування даних. На основі сховища даних побудовано багатовимірний OLAP-куб у SQL Server Analysis Services, який містить ключові виміри та міри для аналізу навчальної успішності студентів. Завершальним етапом роботи є створення аналітичних звітів у SQL Server Reporting Services, що забезпечують підтримку прийняття управлінських рішень у межах діяльності деканату.

Annotatiom

The course project is devoted to the development and analysis of a very large database for the subject area “Student (Dean’s Office)” using Microsoft SQL Server technologies. The work implements a full cycle of information system development, starting from the analysis of the subject area and relational database design, and ending with the construction of a data warehouse, an OLAP cube, and analytical reports.

Within the scope of the project, a database covering the main entities of the dean’s office was designed, including students, groups, teachers, subjects, grades,

hobbies, books, and dormitory accommodation. The database is populated with a large volume of realistic data (over 10,000 students and more than 500,000 grade records), taking into account a five-year study history.

To prepare analytical data, ETL processes were developed using SQL Server Integration Services (SSIS), including data cleansing, transformation, and aggregation. Based on the data warehouse, a multidimensional OLAP cube was built in SQL Server Analysis Services (SSAS), containing key dimensions and measures for analyzing students' academic performance. The final stage of the project involves the creation of analytical reports in SQL Server Reporting Services (SSRS), which provide decision support for managing academic activities within the dean's office.

Зміст

Вступ

У сучасних умовах розвитку інформаційних технологій значна частина управлінських процесів у закладах вищої освіти базується на обробці великих обсягів даних. Деканати університетів щоденно працюють з інформацією про студентів, навчальні групи, дисципліни, викладачів, результати навчання, стипендіальне забезпечення, бібліотечні ресурси та проживання студентів у гуртожитках. Обсяг таких даних постійно зростає, а вимоги до швидкості їх обробки, достовірності та аналітичної глибини стають дедалі вищими.

У зв’язку з цим актуальною є задача створення надвеликої бази даних, яка не лише забезпечує надійне зберігання інформації, але й підтримує ефективний аналіз навчальної діяльності, формування звітності та прийняття управлінських рішень. Сучасні системи керування базами даних повинні поєднувати можливості оперативної обробки транзакцій (OLTP) з аналітичними інструментами (OLAP), що дозволяють працювати з історичними даними та складними агрегатами.

Тема даної курсової роботи — «Розробка та аналіз надвеликої бази даних з використанням технологій Microsoft SQL Server» — є особливо актуальною з огляду на широке застосування платформи Microsoft SQL Server у корпоративних інформаційних системах. Використання таких компонентів, як SQL Server Management Studio, SQL Server Integration Services, SQL Server Analysis Services та SQL Server Reporting Services, дозволяє реалізувати повноцінну аналітичну систему на основі єдиної технологічної платформи.

Предметною областю курсової роботи обрано варіант «Студент (Деканат)», який відображає ключові бізнес-процеси навчального підрозділу закладу вищої освіти. У межах цієї предметної області розглядаються процеси обліку студентів, формування навчальних груп, призначення викладачів і дисциплін, фіксації результатів навчання, контролю заборгованостей по бібліотечних ресурсах, а також управління проживанням студентів у гуртожитках. Особливу увагу приділено аналізу успішності студентів та

формуванню рейтингів, що є важливими показниками якості освітнього процесу.

Метою курсової роботи є розробка повнофункціональної системи керування надвеликою базою даних для деканату з реалізацією ETL-процесів, побудовою сховища даних, багатовимірного OLAP-куба та створенням аналітичних звітів. Для досягнення поставленої мети у роботі необхідно вирішити такі основні завдання:

1. виконати аналіз предметної області та визначити основні сутності і бізнес-правила;
2. спроєктувати реляційну базу даних із дотриманням принципів нормалізації та цілісності даних;
3. згенерувати великі обсяги реалістичних даних відповідно до заданих вимог;
4. реалізувати ETL-процеси з очищенням, трансформацією та завантаженням даних у сховище;
5. побудувати OLAP-куб для багатовимірного аналізу навчальної діяльності;
6. розробити аналітичні звіти для підтримки управлінських рішень.

У ході виконання курсової роботи застосовуються сучасні підходи до проектування баз даних, побудови сховищ даних та аналітичних систем. Результати роботи можуть бути використані як основа для створення реальної інформаційної системи деканату або адаптовані для інших предметних областей з аналогічною структурою даних.

Етап 1. Проектування бази даних

1.1 Аналіз предметної області

Предметною областю даної курсової роботи є діяльність деканату закладу вищої освіти, зокрема облік та аналіз інформації про студентів і пов’язані з ними навчальні та супровідні процеси. В межах обраного варіанту «Студент (Деканат)» розглядається інформаційна система, призначена для зберігання, обробки та аналізу великих обсягів даних, що виникають у процесі навчання студентів протягом кількох років.

У рамках предметної області були виділені основні сутності, що безпосередньо використовуються у роботі деканату: студенти, навчальні групи, викладачі, навчальні предмети, оцінки, хобі студентів, бібліотечні книги та проживання у гуртожитку. Кожна з цих сутностей відображає окремий аспект навчального або адміністративного процесу та має власний набір атрибутів, необхідних для подальшого аналізу.

Основними бізнес-процесами в даній предметній області є:

– формування та облік навчальних груп і закріплення за ними студентів;

– ведення обліку студентів з урахуванням їхніх персональних даних, року вступу та форми навчання;

– фіксація результатів навчання у вигляді оцінок за навчальні дисципліни;

– облік викладачів і дисциплін, які вони викладають;

– контроль використання бібліотечних ресурсів та виявлення заборгованостей за книгами;

– облік проживання студентів у гуртожитку протягом навчального періоду.

На основі аналізу предметної області були визначені функціональні вимоги до системи. Система повинна забезпечувати збереження великих обсягів структурованих даних, підтримувати історію навчання студентів за п’ять років, а також надавати можливість виконання аналітичних запитів для формування звітів. Зокрема, система має забезпечувати отримання списків

студентів по групах, побудову рейтингів студентів за заданими параметрами, аналіз заборгованостей по бібліотечних книгах, формування довідок про успішність студентів та аналіз навчальної успішності за предметами і викладачами.

У ході аналізу також були сформульовані основні бізнес-правила та обмеження, які враховувалися під час проектування бази даних. Кількість студентів у системі повинна становити не менше 10 000 осіб, а кількість записів про оцінки — не менше 500 000. Оцінки зберігаються у числовому діапазоні від 0 до 100 балів з подальшою можливістю їх трансформації в інших підсистемах. Усі дати, що зберігаються в системі, обмежені реалістичними часовими рамками та не перевищують кінець 2025 року. Також було забезпечено логічну узгодженість даних між сутностями, зокрема відповідність навчальних груп факультетам та спеціальностям.

Проведений аналіз предметної області став основою для подальшого проектування структури бази даних та реалізації наступних етапів курсової роботи.

1.2 Концептуальне проектування

На етапі концептуального проєктування бази даних було виконано узагальнення результатів аналізу предметної області та побудовано концептуальну модель даних у вигляді ER-діаграми.

У процесі проєктування були визначені основні сутності, що відповідають об’єктам предметної області деканату. До них належать: Студент, Група, Предмет, Викладач, Оцінка, Хобі, Книга, Кімната гуртожитку, а також допоміжні сутності для реалізації зв’язків між ними. Кожна сутність має унікальний ідентифікатор та набір атрибутів, необхідних для зберігання та подальшого аналізу даних.

Сутність «Група» описує навчальні групи та містить атрибути, що характеризують факультет, спеціальність, рік набору та форму навчання.

Сутність «Студент» відображає основну інформацію про студентів і пов’язана з навчальною групою, до якої належить студент.

Сутності «Викладач» та «Предмет» використовуються для опису навчального процесу та формування результатів успішності.

Сутність «Оцінка» призначена для зберігання результатів навчання студентів з конкретних предметів.

Сутності «Хобі» та «Книга» використовуються для збереження додаткової інформації про студентів і бібліотечні ресурси.

Сутність «Кімната гуртожитку» описує житлові приміщення, у яких можуть проживати студенти.

Між сутностями були визначені логічні зв’язки відповідно до бізнес-процесів предметної області. Зокрема, між сутностями «Група» та «Студент» встановлено зв’язок типу один-до-багатьох, оскільки одна навчальна група може містити багато студентів, але кожен студент належить лише до однієї групи. Між сутностями «Студент» та «Оцінка» також реалізовано зв’язок один-до-багатьох, оскільки кожен студент може мати багато оцінок за різні предмети.

Зв’язок між сутностями «Предмет» та «Викладач» реалізовано через навчальний процес, у межах якого один викладач може викладати декілька предметів, а один предмет може викладатися різними викладачами в різні навчальні періоди. Зв’язок між сутностями «Студент» та «Хобі» має тип багато-до-багатьох, оскільки один студент може мати кілька хобі, а одне хобі може бути притаманним багатьом студентам. Аналогічний тип зв’язку реалізовано між сутностями «Студент» та «Книга», що дозволяє враховувати багаторазове користування бібліотечними ресурсами різними студентами.

Кардинальність усіх зв’язків була визначена з урахуванням реальних обмежень предметної області та подальшої реалізації у реляційній базі даних. Побудована ER-діаграма стала основою для переходу до наступного етапу — логічного та фізичного проєктування бази даних, а також забезпечила цілісність і узгодженість структури даних у подальших етапах курсової роботи.

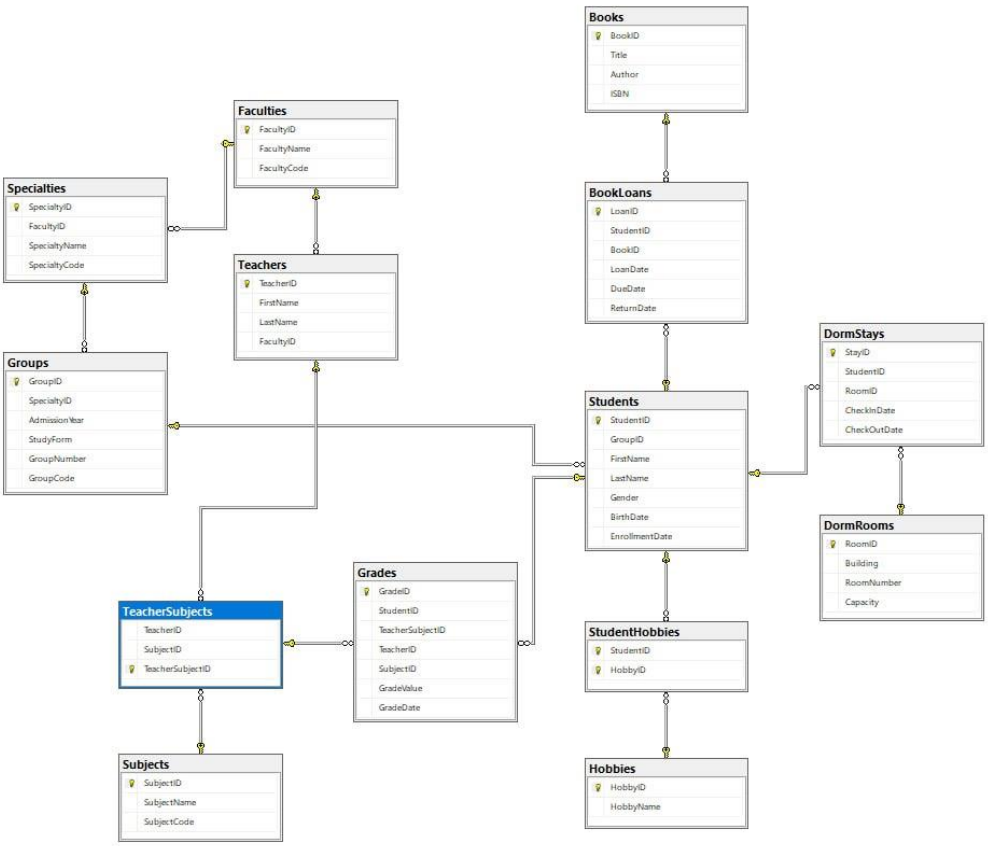


Рис 1. 1 ER-діаграма

1.3 Логічне проектування

На етапі логічного проєктування було виконано перехід від концептуальної моделі до реляційної структури бази даних з урахуванням вимог предметної області та подальшого аналітичного використання даних. Проєктування виконувалося з дотриманням принципів нормалізації з метою усунення надмірності даних та забезпечення їх цілісності.

У процесі проєктування структура бази даних була нормалізована щонайменше до третьої нормальної форми (3НФ). Усі таблиці містять атомарні атрибути, не мають повторюваних груп і залежать від первинного ключа. Часткові та транзитивні залежності були усунуті шляхом винесення окремих сутностей у самостійні таблиці. Зокрема, інформація про студентів, групи, викладачів, предмети, оцінки, хобі, книги та проживання у гуртожитку зберігається в окремих таблицях, що відповідає вимогам 3НФ.

Денормалізація на етапі логічного проєктування не застосовувалася, оскільки основною метою цього етапу було створення коректної та гнучкої

реляційної моделі для подальшого наповнення великими обсягами даних. Оптимізація доступу до даних та агрегація показників передбачені на наступних етапах курсової роботи під час побудови сховища даних та OLAP-куба.

Для кожної таблиці були визначені первинні ключі, які однозначно ідентифікують записи. Як первинні ключі використовуються сурогатні ідентифікатори типу INT або BIGINT, що забезпечує ефективну індексацію та зручність при роботі з великими обсягами даних. Наприклад, для таблиць студентів, груп, предметів та викладачів використано окремі ідентифікатори, що не залежать від бізнес-атрибутів.

Між таблицями були визначені зовнішні ключі, які реалізують зв’язки між сутностями та забезпечують цілісність даних. Зокрема, таблиця студентів містить зовнішній ключ на таблицю груп, таблиця оцінок пов’язана зі студентами та предметами, а таблиці, що реалізують зв’язки багато-до-багатьох, містять зовнішні ключі на відповідні основні сутності. Наявність зовнішніх ключів унеможливлює появу некоректних або неузгоджених даних.

З метою оптимізації виконання запитів були спроєктовані індекси на ключових атрибутах. Первинні ключі автоматично індексуються, а додаткові індекси створені на зовнішніх ключах та полях, які найчастіше використовуються у фільтрації, групуванні та з’єднанні таблиць. Такий підхід дозволяє підвищити продуктивність запитів, що виконуються над великими обсягами даних, зокрема під час формування аналітичних звітів.

У результаті логічного проєктування була сформована реляційна модель бази даних, яка є цілісною, нормалізованою та готовою до фізичної реалізації і наповнення даними на наступних етапах курсової роботи.

1.4 Фізичне проектування

На етапі фізичного проєктування було виконано безпосередню реалізацію структури бази даних у середовищі Microsoft SQL Server. Фізична модель бази даних була створена на основі логічної моделі з урахуванням

вимог до цілісності даних, продуктивності та подальшого наповнення великими обсягами інформації.

Для реалізації фізичної структури бази даних були розроблені SQL-скрипти створення таблиць, які визначають набір атрибутів, типи даних та первинні ключі для кожної сутності. Усі таблиці створені з використанням відповідних типів даних, що забезпечують коректне зберігання числової, текстової та датованої інформації. Структура таблиць відповідає спроєктованій логічній моделі та забезпечує можливість масштабування бази даних.

З метою забезпечення цілісності даних у базі були реалізовані обмеження цілісності. Для контролю допустимих значень атрибутів використовувалися обмеження типу CHECK, зокрема для числових полів та дат, що дозволяє зберігати лише реалістичні значення. Обмеження UNIQUE застосовувалися до атрибутів, які повинні мати унікальні значення в межах таблиці. Для підтримки зв’язків між сутностями були створені обмеження FOREIGN KEY, які гарантують узгодженість даних між пов’язаними таблицями та запобігають появі некоректних посилань.

Створення тригерів для реалізації складних бізнес-правил у межах курсової роботи не застосовувалося, оскільки основні правила предметної області були реалізовані за допомогою обмежень цілісності та логіки подальших ETL-процесів. Такий підхід дозволив спростити фізичну модель бази даних і уникнути додаткового навантаження на систему під час масового завантаження даних.

Для підтримки роботи з даними та подальшого аналізу були створені збережені процедури та користувацькі функції, які використовуються для перевірки коректності даних, виконання типових операцій вибірки та попереднього аналізу інформації. Застосування збережених процедур і функцій дозволяє централізувати бізнес-логіку на рівні бази даних та підвищити зручність використання системи.

У результаті фізичного проєктування була створена повноцінна реляційна база даних, готова до наповнення великими обсягами даних та подальшої реалізації аналітичних і звітних компонентів курсової роботи.

Етап 2. Генерація та наповнення даних

2.1 Підготовка до генерації даних

На даному етапі курсової роботи було виконано підготовку до наповнення бази даних великими обсягами інформації з урахуванням вимог до реалістичності та масштабності даних. Основною метою цього етапу було забезпечення коректної генерації даних, які максимально наближені до реальних умов функціонування інформаційної системи деканату.

Для генерації даних було обрано інструмент Redgate SQL Data Generator, який інтегрується безпосередньо з Microsoft SQL Server та дозволяє генерувати великі обсяги даних з урахуванням структури таблиць, обмежень цілісності та зв’язків між ними. Використання Redgate SQL Data Generator дозволило виконувати генерацію даних поетапно, у правильному порядку залежностей між таблицями, що є критично важливим для реляційної бази даних з великою кількістю зовнішніх ключів. Інші інструменти генерації даних, такі як Mockaroo або власні SQL-скрипти, не використовувалися, оскільки вони не забезпечують такого рівня контролю над зв’язками між таблицями та обмеженнями цілісності.

Під час підготовки до генерації було проаналізовано вимоги до реалістичності даних відповідно до технічного завдання. Зокрема, кількість студентів у базі даних повинна становити не менше 10 000, а кількість записів оцінок — не менше 500 000. Для забезпечення реалістичності часових характеристик усі дати, що генеруються в системі, були обмежені п’ятирічним періодом навчання та не перевищують кінець 2025 року. Вік студентів задавався у реалістичних межах, а значення оцінок — у діапазоні від 0 до 100 балів, що дозволяє у подальшому виконувати трансформації та аналітичні розрахунки.

Окрему увагу було приділено узгодженості довідкових даних. Зокрема, для навчальних груп було забезпечено відповідність факультетів та спеціальностей шляхом використання окремого довідника спеціальностей, прив’язаних до конкретних факультетів. Такий підхід дозволив уникнути логічних помилок у даних, наприклад, появи спеціальностей, які не відповідають обраному факультету.

На основі аналізу структури бази даних було визначено стратегію розподілу даних між таблицями. Генерація виконувалася у декілька етапів: спочатку наповнювалися довідкові таблиці, після чого генерувалися основні сутності, зокрема навчальні групи та студенти. На завершальному етапі створювалися фактологічні дані, такі як записи про оцінки та інші події, що забезпечило коректність зовнішніх ключів і відповідність бізнес-правилам предметної області.

2.2 Генерація великих обсягів даних

На даному етапі курсової роботи було виконано безпосередню генерацію великих обсягів даних для наповнення розробленої бази даних. Генерація здійснювалася з використанням інструменту Redgate SQL Data Generator відповідно до визначеної на попередньому етапі стратегії розподілу даних та з урахуванням усіх обмежень цілісності.

Під час генерації основна увага приділялася виконанню кількісних вимог технічного завдання. В основних таблицях фактів, зокрема таблиці оцінок, було згенеровано не менше 500 000 записів, що забезпечує достатній обсяг даних для подальшого аналітичного оброблення. Довідникові таблиці, які використовуються для опису факультетів, спеціальностей, навчальних груп, предметів та інших допоміжних сутностей, були наповнені значною кількістю записів, що у сукупності перевищує 100 000 рядків.

Процес генерації даних виконувався поетапно з урахуванням залежностей між таблицями. Спочатку наповнювалися довідникові таблиці, після чого генерувалися основні сутності, такі як навчальні групи та студенти. На завершальному етапі створювалися фактологічні дані, зокрема записи про

оцінки студентів за навчальні дисципліни. Такий підхід дозволив уникнути порушення зовнішніх ключів і забезпечити коректне формування зв’язків між сутностями.

Для забезпечення реалістичності даних генерація виконувалася з урахуванням бізнес-логіки предметної області. Значення оцінок задавалися у числовому діапазоні від 0 до 100 балів, що відповідає реальній системі оцінювання та дозволяє у подальшому виконувати їх трансформацію. Дані про студентів генерувалися з урахуванням реалістичного віку та періоду навчання, а всі дати обмежувалися п’ятирічним часовим інтервалом і не перевищували кінець 2025 року. Для навчальних груп було забезпечено логічну відповідність між факультетами та спеціальностями.

У процесі генерації було дотримано всіх обмежень цілісності, визначених на етапі фізичного проєктування бази даних. Обмеження типу CHECK, UNIQUE та FOREIGN KEY не порушувалися, що підтверджує коректність налаштувань генераторів та послідовність наповнення таблиць. Після завершення генерації виконувалася перевірка кількості записів і логічної узгодженості даних за допомогою контрольних SQL-запитів.

2.3 Налаштування генератора даних

На даному етапі курсової роботи було виконано детальне налаштування генератора даних для коректного наповнення розробленої бази даних великими обсягами інформації. Налаштування здійснювалося з використанням інструменту Redgate SQL Data Generator з урахуванням структури таблиць, обмежень цілісності та бізнес-логіки предметної області.

Для кожної таблиці бази даних були окремо налаштовані правила генерації даних. Для довідникових таблиць використовувалися фіксовані списки значень або регулярні шаблони, що дозволило отримати контрольовані та логічно узгоджені дані. Для основних таблиць застосовувалися генератори числових, текстових і датованих значень з обмеженнями діапазонів, які відповідають реальним умовам функціонування системи деканату. Особлива

увага приділялася полям, що беруть участь у зовнішніх ключах, для яких використовувалися механізми посилання на вже згенеровані дані.

Зв’язність даних між таблицями забезпечувалася шляхом поетапної генерації та коректного налаштування зовнішніх ключів. Генерація виконувалася у визначеній послідовності: спочатку наповнювалися довідникові таблиці, після чого створювалися записи основних сутностей, і лише на завершальному етапі — фактологічні дані. Такий підхід гарантував відсутність порушень обмежень FOREIGN KEY і забезпечив логічну узгодженість між пов’язаними таблицями.

Під час налаштування генератора також було враховано часові періоди зберігання даних. Для атрибутів типу дати встановлювалися реалістичні діапазони значень, що охоплюють період навчання тривалістю не менше трьох–п’яти років. Усі дати обмежувалися кінцем 2025 року, що відповідає вимогам технічного завдання та дозволяє аналізувати історичні дані у динаміці.

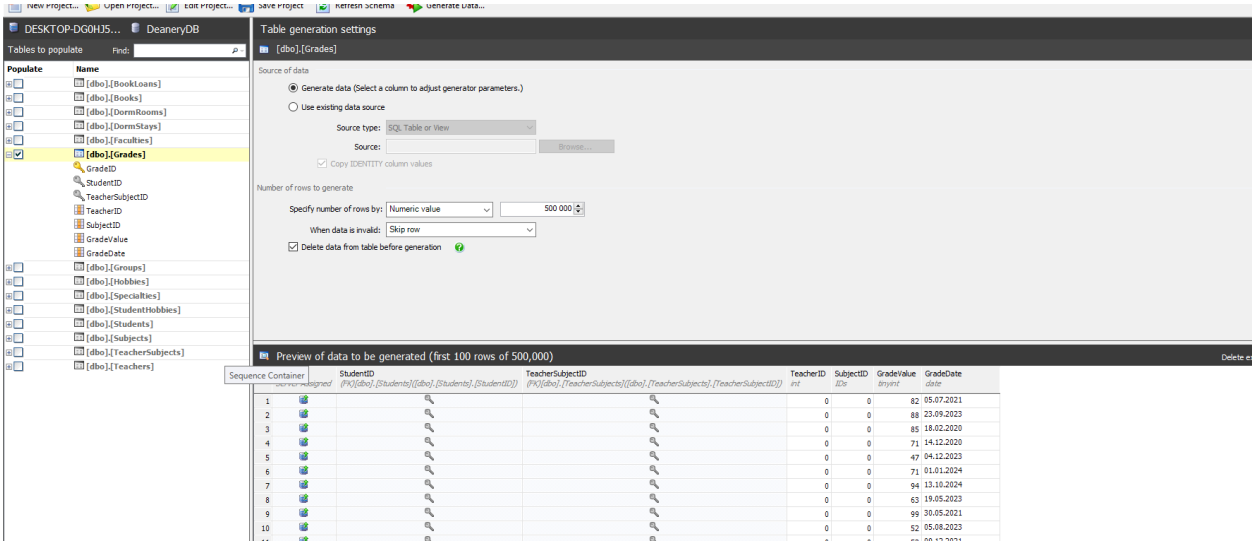


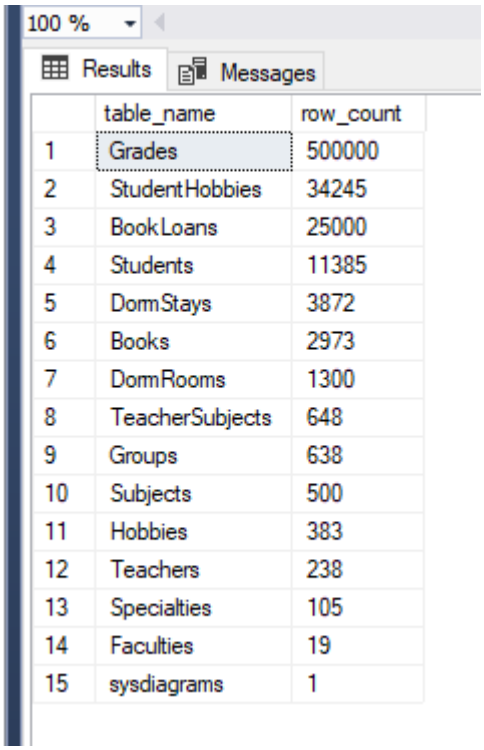
Рис 2.3 Генерація даних в Red Gate

Після завершення генерації даних було створено резервні копії бази даних, що дозволяє зберегти результати генерації та забезпечити можливість відновлення даних у разі необхідності. Наявність резервної копії також спрощує повторне використання згенерованих даних на наступних етапах курсової роботи без повторної генерації.

2.4 Верифікація даних

На даному етапі курсової роботи було виконано верифікацію згенерованих даних з метою перевірки їх коректності, повноти та відповідності вимогам технічного завдання. Верифікація здійснювалася безпосередньо на рівні бази даних за допомогою SQL-запитів у середовищі Microsoft SQL Server Management Studio.

Першим кроком верифікації стала перевірка кількості згенерованих записів у таблицях бази даних. За допомогою агрегатних SQL-запитів було підтверджено, що кількість записів в основних таблицях фактів відповідає заданим вимогам і перевищує мінімальне значення у 500 000 рядків. Також було перевірено наповнення довідникових таблиць, у яких загальна кількість записів перевищує 100 000, що забезпечує достатню різноманітність і деталізацію даних для подальшого аналізу.


100 %

Results | Messages

	table_name	row_count
1	Grades	500000
2	StudentHobbies	34245
3	BookLoans	25000
4	Students	11385
5	DormStays	3872
6	Books	2973
7	DormRooms	1300
8	TeacherSubjects	648
9	Groups	638
10	Subjects	500
11	Hobbies	383
12	Teachers	238
13	Specialties	105
14	Faculties	19
15	sysdiagrams	1

Рис 2.5 Сгенеровані дані

Далі було виконано аналіз якості та реалістичності даних. Перевірялися діапазони числових значень, зокрема оцінок студентів, які знаходяться у межах від 0 до 100 балів. Окрему увагу було приділено коректності дат: усі часові атрибути відповідають заданому періоду навчання тривалістю кілька років та

не перевищують кінець 2025 року. Також було перевірено логічну узгодженість довідкових даних, зокрема відповідність факультетів та спеціальностей у навчальних групах, що підтвердило коректність обраної стратегії генерації.

Завершальним етапом верифікації стало тестування продуктивності запитів на великих обсягах даних. Було виконано типові запити вибірки, агрегації та з'єднання таблиць, які використовуються під час формування звітів та аналітичних вибірок. Отримані результати показали, що запити виконуються коректно і з прийнятним часом відгуку, що підтверджує доцільність використання спроєктованої структури бази даних та індексів для роботи з великими обсягами інформації.

Таким чином, проведена верифікація підтвердила, що згенеровані дані є повними, реалістичними та придатними для подальшої реалізації ETL-процесів і аналітичних компонентів курсової роботи.

ЕТАП 3. Реалізація ETL-процесів (SQL Server Integration Services)

3.1. Створення проекту SSIS

Для реалізації процесів вилучення, трансформації та завантаження даних (ETL) я обрав інструментарій Microsoft SQL Server Integration Services (SSIS), оскільки він забезпечує високу продуктивність при обробці великих масивів даних (понад 500 000 записів) та надає гнучкі можливості для очищення даних.

Встановлення SQL Server Data Tools (SSDT)

У якості середовища розробки я використовував Microsoft Visual Studio 2022. Для роботи з SSIS я переконався у наявності та коректному встановленні розширення "SQL Server Integration Services Projects". Це дозволило розблокувати специфічні для Business Intelligence шаблони проектів та панель інструментів SSIS Toolbox, яка містить необхідні компоненти для побудови потоків даних (Data Flow) та керування процесами (Control Flow).

Створення нового проекту Integration Services

У середовищі Visual Studio мною було створено новий проект типу Integration Services Project. Я обрав саме цей тип проекту, оскільки він дозволяє

створювати складні пакети (.dtsx), керувати параметрами проекту та налаштовувати рівні захисту даних. Структура проекту була організована таким чином, щоб розділити логіку завантаження вимірів та фактів у різні пакети для зручності підтримки та налагодження.

Налаштування з'єднань з джерелами даних

Ключовим етапом підготовки стала конфігурація менеджерів з'єднань (Connection Managers), які забезпечують зв'язок між ETL-пакетами та базами даних. Я використав провайдер OLE DB для забезпечення максимальної швидкодії. Мною було створено та налаштовано два основних з'єднання:

З'єднання з джерелом (Source Connection) налаштовано на вихідну OLTP-базу даних, наповнену тестовими даними (студенти, оцінки, книги) за допомогою інструменту RedGate.

З'єднання з призначенням (Destination Connection) налаштовано на створену мною базу даних сховища DeaneryDWH, яка має структуру схеми "Зірка".

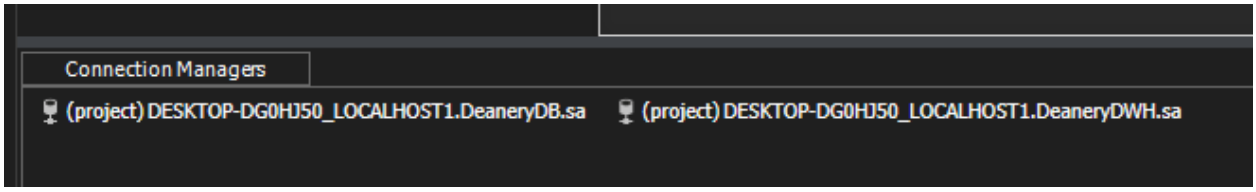


Рис 3.1 З'єднання

Коректність з'єднань була перевірена за допомогою функції "Test Connection", що підтвердило доступність обох баз даних для операцій читання та запису.

3.2. Проектування Data Warehouse

Архитектура DW: Схема зірки (Star Schema)

Для проектування сховища даних DeaneryDWH я обрав архітектуру "Схема зірки" (Star Schema). Цей вибір обумовлений необхідністю забезпечення високої швидкодії аналітичних запитів та простоти структури для подальшої побудови OLAP-кубів. На відміну від нормалізованої схеми "Сніжинка", схема "Зірки" дозволяє мінімізувати кількість операцій з'єднання (JOIN) при формуванні звітів, денормалізуючи ієрархічні дані безпосередньо

у таблиці вимірів. Центром схеми є таблиці фактів, що містять кількісні показники (міри) та зовнішні ключі, оточені таблицями вимірів, які містять описові атрибути.

Таблиці фактів (Fact Tables)

Мною було спроектовано та створено 3 таблиці фактів (що перевищує мінімальну вимогу у 2 таблиці), які покривають ключові бізнес-процеси деканату:

FactGrades (Успішність): Основна таблиця фактів, що містить понад 500 000 записів. Вона зберігає інформацію про кожну отриману оцінку.

Ключі: StudentKey, SubjectKey, TeacherKey, DateKey.

Міри: GradeValue (бал), IsPassing (індикатор успішного складання).

FactLibraryLoans (Бібліотечні видачі): Відображає процес обігу книг.

Ключі: StudentKey, BookKey, LoanDateKey, DueDateKey, ReturnDateKey.

Міри: LoanDurationDays (тривалість користування), OverdueDays (кількість днів прострочення), IsOverdue (факт боргу).

FactDormStays (Проживання в гуртожитку): Додаткова таблиця для аналізу поселення.

Ключі: StudentKey, DormRoomKey, CheckInDateKey.

Міри: StayDurationDays, IsActive (поточний статус проживання).

Таблиці вимірів (Dimension Tables)

Для забезпечення повноти аналізу я розробив 6 таблиць вимірів (вимога — мінімум 5), використовуючи сурогатні ключі (Surrogate Keys) для однозначної ідентифікації записів:

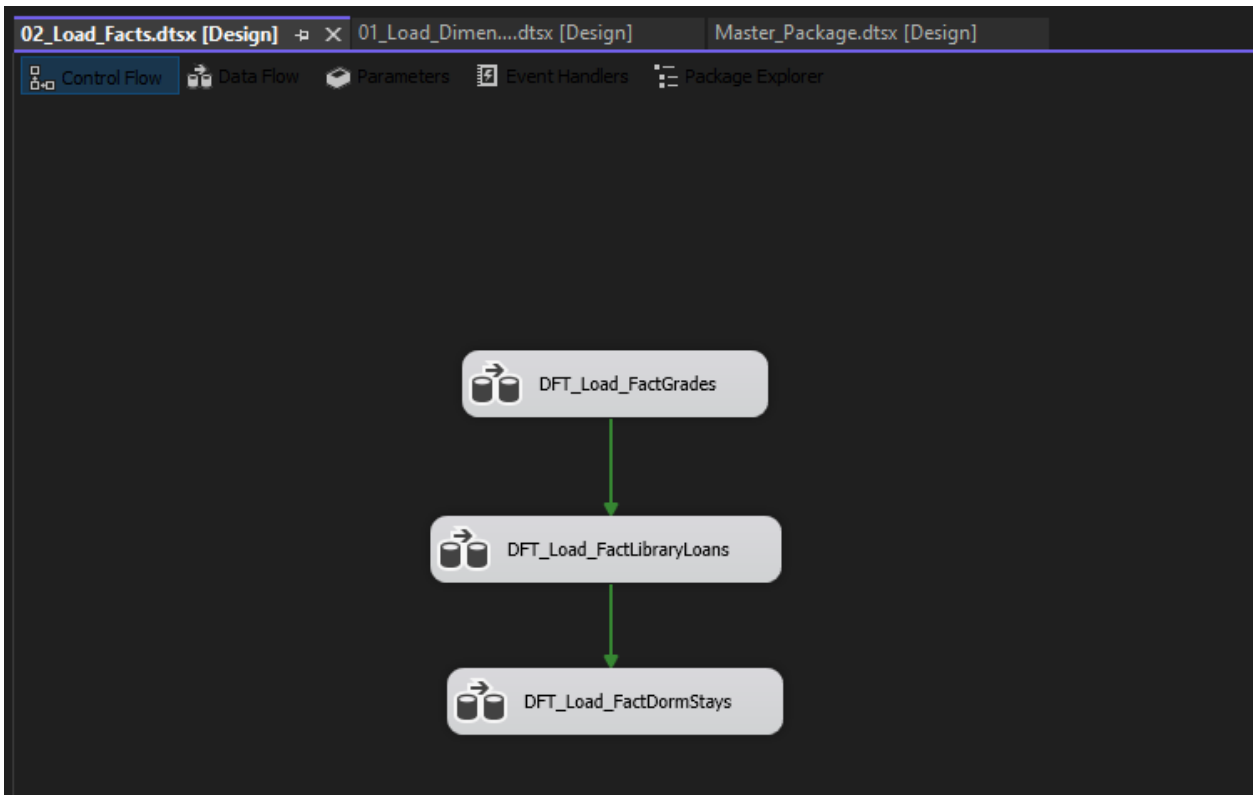


Рис 3.2.1 Завантаження фактів

DimStudents: Містить повну інформацію про студентів, включно з денормалізованими даними про групу, спеціальність та факультет.

DimDate: Критично важливий часовий вимір, згенерований для діапазону 2015–2040 років, що містить атрибути року, кварталу, місяця, дня тижня та ознаки вихідного дня.

DimSubjects: Довідник навчальних дисциплін.

DimTeachers: Інформація про викладачів та їх приналежність до кафедри/факультету.

DimBooks: Атрибути книг (автор, назва, ISBN).

DimDormRooms: Характеристики житлового фонду (номер кімнати, корпус, місткість).

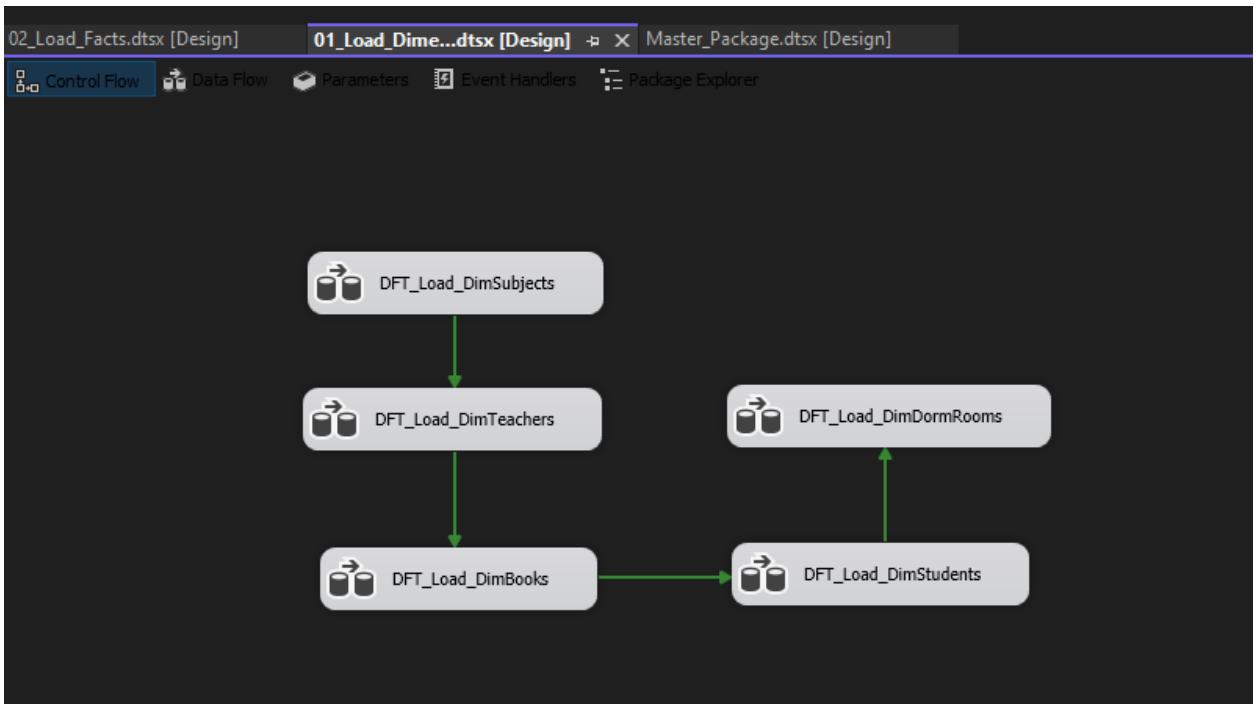


Рис 3.2.1 Завантаження вимірів

Реалізація SCD (Slowly Changing Dimensions)

Однією з найскладніших частин проектування стала реалізація механізму відстеження історичних змін. Для виміру DimStudents я застосував гібридний підхід:

SCD Type 1 (Перезапис): Використовується для виправлення помилок у стабільних атрибутах (наприклад, виправлення помилки в імені FullName або статі Gender). Історія змін цих полів не зберігається.

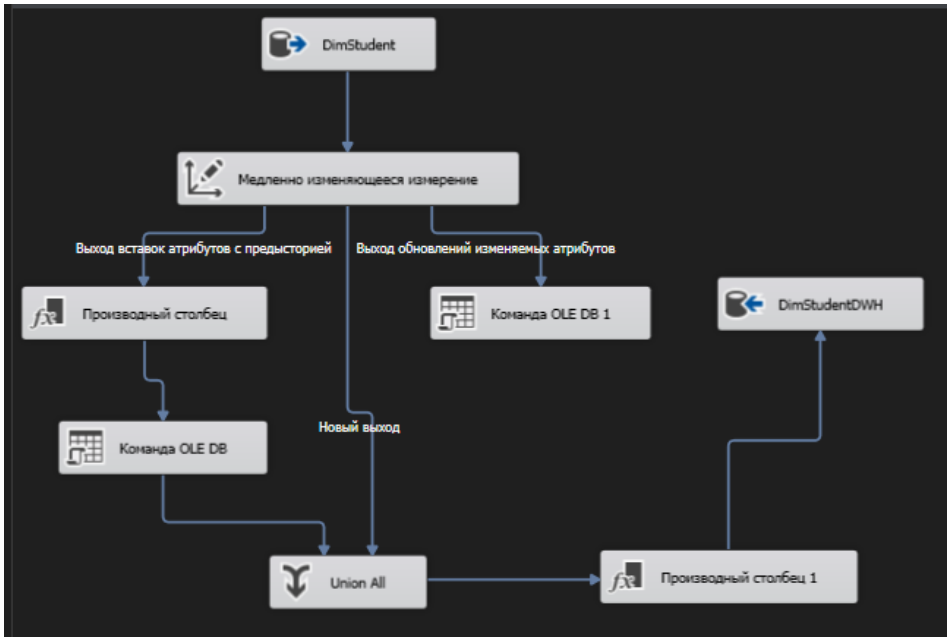


Рис 3.2.3 Реалізація SCD

3.3. Розробка ETL-пакетів

Створення Data Flow Task

Для кожного логічного блоку даних (завантаження довідників, завантаження фактів) я створив окремі задачі потоку даних (Data Flow Tasks) у середовищі Control Flow. Це дозволило розпаралелити процеси та структурувати логіку виконання.

Налаштування OLE DB Source

В якості компонента вилучення даних я використовував OLE DB Source. Залежно від складності даних, я застосував два підходи:

Table or View: Для простих довідників (наприклад, Books, Subjects) дані витягувалися безпосередньо з таблиць.

SQL Command: Для таблиць фактів (FactGrades, FactLibraryLoans) та складних вимірів я писав оптимізовані SQL-запити безпосередньо у джерелі.

Мною було реалізовано 5 типів трансформацій, необхідних для очищення та підготовки даних згідно з бізнес-правилами:

1. Data Cleansing (Очищення даних) — компонент Conditional Split

Для забезпечення якості даних я застосував компонент Conditional Split у двох сценаріях:

Фільтрація оцінок: Було створено правило для відсіювання некоректних значень оцінок (менше 1 або більше 100). Записи, що не пройшли перевірку, перенаправлялися у потік "Bad Data" і ігнорувалися.

Логічна перевірка дат: При завантаженні бібліотечних позик я реалізував фільтр для відкидання записів, де дата повернення була ранішою за дату видачі (від'ємна тривалість користування), що є логічною помилкою генерації даних.

2. Data Conversion (Конвертація даних) — компонент Data Conversion

При завантаженні виміру DimSubjects виникла необхідність приведення типів даних. Я використав цей компонент для конвертації рядкових даних

(DT_STR) у формат Unicode (DT_WSTR), щоб уникнути конфліктів типів при записі в цільову базу даних.

3. Derived Column (Похідні колонки) — компонент Derived Column

Цей компонент використовувався для створення нових бізнес-атрибутів, яких не було у вихідній базі:

Формування ПІБ: Для таблиці DimTeachers я створив похідну колонку FullName, об'єднавши поля FirstName та LastName через пробіл.

Бізнес-логіка: Для таблиці FactGrades було створено обчислюване поле IsPassing (Успішність), яке присвоює значення 1, якщо оцінка > 60, та 0, якщо менше.

4. Lookup (Пошук довідників) — компонент Lookup

Це найбільш вживана трансформація у моєму проекті. Вона використовувалася для заміни бізнес-ключів (ID) з вихідної системи на сурогатні ключі (Key) зі сховища даних:

Я налаштував зіставлення для отримання SubjectKey, TeacherKey, BookKey.

Для обробки записів, що не знайдені (наприклад, якщо факультет викладача не вказано), я використовував режим "Ignore failure" або перенаправляв рядки (залежно від критичності).

Особливість: Для пошуку студентів (StudentKey) я використовував SQL-запит всередині Lookup (WHERE IsCurrent = 1), щоб отримати ID саме актуального запису студента, враховуючи історію змін.

5. Slowly Changing Dimension (SCD) — компонент SCD Wizard

Для виміру DimStudents я реалізував складну логіку керування історичністю (SCD Type 1). Майстер SCD автоматично згенерував необхідні трансформації (Conditional Split, OLE DB Command, Derived Column) для перевірки, чи змінилися дані студента (група, спеціальність).

Створення OLE DB Destination

Фінальним етапом кожного потоку даних стало завантаження у таблиці сховища DeaneryDWH. Я налаштував зіставлення (Mappings) між трансформованими потоками та цільовими таблицями.

Налаштування режимів завантаження

Мною була реалізована стратегія Full Load (Повне перезавантаження) для демонстрації цілісності процесу. Перед кожним запуском ETL спеціальний SQL-скрипт (у Master Package) очищає таблиці фактів (DELETE) та скидає лічильники ідентифікаторів (DBCC CHECKIDENT), що гарантує відсутність дублікатів та коректність даних при багаторазовому запуску.

3.4. Додаткові компоненти SSIS

Для оркестрації процесу завантаження, керування послідовністю виконання та забезпечення стабільності роботи системи я створив керуючий пакет **00_Master_Package.dtsx**, у якому реалізував наступні компоненти:

Execute SQL Task (Виконання SQL-запитів)

Цей компонент я використовував для виконання підготовчих та службових операцій безпосередньо в базі даних:

Очищення DWH: На початку роботи пакету задача **SQL_Clean_DWH** виконує скрипт видалення даних (DELETE) з таблиць фактів та вимірів, а також скидає лічильники ідентифікаторів (DBCC CHECKIDENT). Це забезпечує "чистий старт" для демонстрації повного завантаження.

Логування: Задача **SQL_Log_Start** виконує INSERT у таблицю аудиту **Sys_AuditLog**, фіксуючи час початку виконання ETL-процесу.

Оновлення статистики: Всередині циклу я використовую цей компонент для виконання команди UPDATE STATISTICS для кожної таблиці, що оптимізує роботу бази даних після завантаження.

Sequence Container (Контейнер послідовності)

Для логічної організації робочого потоку я використав Sequence Container (**Container_Run_ETL**). Він об'єднує задачі запуску дочірніх пакетів (Execute Package Task): завантаження вимірів (**01_Load_Dimensions**) та завантаження фактів (**02_Load_Facts**). Використання контейнера дозволяє

візуально відокремити основний блок виконання ETL від підготовчих операцій та пост-обробки, а також керувати властивостями всієї групи завдань одночасно.

For Each Loop Container (Цикл для масової обробки)

Я реалізував цикл For Each Loop (Update Statistics Loop) для автоматизації обслуговування бази даних.

Налаштування: Контейнер використовує колекцію Foreach Item Enumerator, де я задав список ключових таблиць сховища (FactGrades, DimStudents, FactLibraryLoans тощо).

Змінні: На кожній ітерації ім'я поточної таблиці передається у змінну SSIS User::TableName.

Дія: Всередині циклу динамічно формується SQL-запит для оновлення статистики саме тієї таблиці, яка зараз обробляється. Це дозволяє легко додавати нові таблиці в обслуговування без зміни коду SQL.

Script Task (Завдання скрипта)

Для реалізації бізнес-логіки, що виходить за межі стандартних компонентів, я використав Script Task. Мною був написаний код мовою C#, який виконується після успішного завершення всіх етапів завантаження. Скрипт створює текстовий файл-звіт (**ETL_Status.txt**) на локальному диску, записуючи туди повідомлення про успішне завершення та точний час. Це імітує відправку повідомлення адміністратору системи.

Event Handlers (Обробники подій)

Для підвищення надійності системи я налаштував механізм обробки помилок.

На рівні пакету **00_Master_Package** я створив обробник події OnError.

У разі виникнення будь-якої критичної помилки під час виконання пакету, автоматично запускається спеціальна задача Execute SQL Task, яка записує повідомлення про помилку та статус ERROR у таблицю **Sys_AuditLog**. Це дозволяє швидко діагностувати проблеми без необхідності аналізувати системні логи сервера.

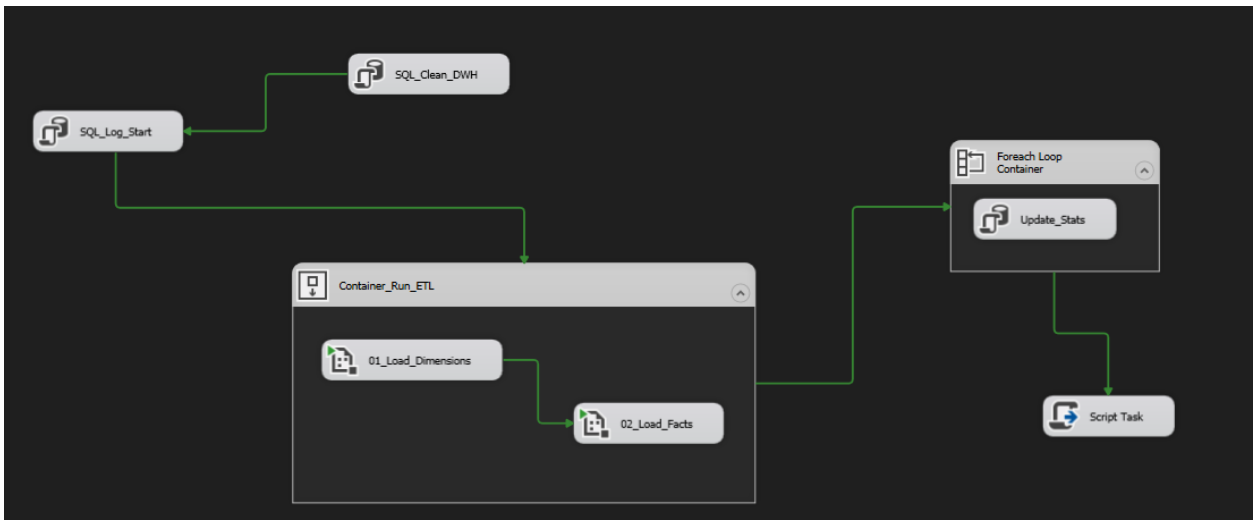


Рис 3.4.1 MasterPackage з додатковим функціоналом

3.5. Контроль якості ETL

Для забезпечення надійності роботи сховища даних та моніторингу виконання процесів завантаження мною була розроблена комплексна система контролю якості.

Створення таблиць аудиту ETL-процесів

Перед початком розробки пакетів я створив у базі даних DeaneryDWH спеціальну службову таблицю dbo.Sys_AuditLog. Ця таблиця виступає єдиним журналом для реєстрації подій системи і має наступну структуру:

PackageName — назва пакету, що виконується;

TaskName — назва конкретної задачі;

Message — текст повідомлення (інформаційний або текст помилки);

LogDate — автоматична мітка часу події;

Status — статус операції (STARTED, SUCCESS, ERROR).

Логування виконання пакетів

Мною було налаштовано автоматичне логування ключових етапів роботи системи через компонент Execute SQL Task у майстер-пакеті:

Фіксація старту: При запуску **00_Master_Package** до таблиці аудиту додається запис зі статусом STARTED. Це дозволяє адміністратору бачити, коли саме почалося оновлення сховища.

Фіксація помилок: Завдяки налаштованим Event Handlers, у разі аварійної зупинки пакету до логу автоматично додається запис зі статусом ERROR та описом проблеми.

Виявлення та обробка помилкових записів

Я реалізував механізм перевірки якості даних під час виконання Data Flow за допомогою компонента Conditional Split:

Бізнес-правила: Було виявлено проблему некоректних дат у джерелі (коли дата повернення книги передувала даті видачі — "помилка мандрівника у часі"). Я налаштував умову фільтрації LoanDurationDays >= 0.

Ізоляція помилок: Записи, що не відповідають умовам якості, автоматично направляються в окремий вихідний потік (Bad Data) і не потрапляють до цільових таблиць фактів, що гарантує чистоту даних у кубі.

Звіти про результати обробки

Для інформування про результати завантаження я використав Script Task із кодом на C#. Після успішного виконання всіх пакетів скрипт генерує або доповнює текстовий файл-звіт (**ETL_Status.txt**) на сервері. У звіті фіксується успішне завершення повного циклу ETL та точний час закінчення операції. Це дозволяє швидко перевірити актуальність даних у сховищі без необхідності виконувати SQL-запити до бази.

Етап 4. Побудова OLAP-куба (SQL Server Analysis Services)

4.1. Створення проекту SSAS

Створення нового проекту Analysis Services Multidimensional

Для розробки багатовимірної аналітичної моделі я використав середовище Microsoft Visual Studio 2022. Мною був створений новий проект типу Analysis Services Multidimensional and Data Mining Project. Я обрав саме багатовимірний (Multidimensional) режим роботи, а не табличний (Tabular), оскільки він забезпечує класичну структуру OLAP-кубів (MOLAP) з розширеними можливостями агрегації та використання мови запитів MDX, що є вимогою курсової роботи.

Налаштування з'єднання з Data Warehouse

Першим кроком у проекті я створив об'єкт Data Source (Джерело даних). Я налаштував рядок підключення (Connection String) до локального екземпляра бази даних DeaneryDWH, яку ми спроектували та наповнили на етапі ETL. Коректність з'єднання була підтверджена успішним проходженням тесту підключення, що гарантує доступ SSAS до таблиць фактів та вимірів.

Створення Data Source View (DSV)

Для логічного представлення даних я створив Data Source View (DSV) під назвою **DeaneryDWH**. Це проміжний шар абстракції між фізичною базою даних та структурою куба.

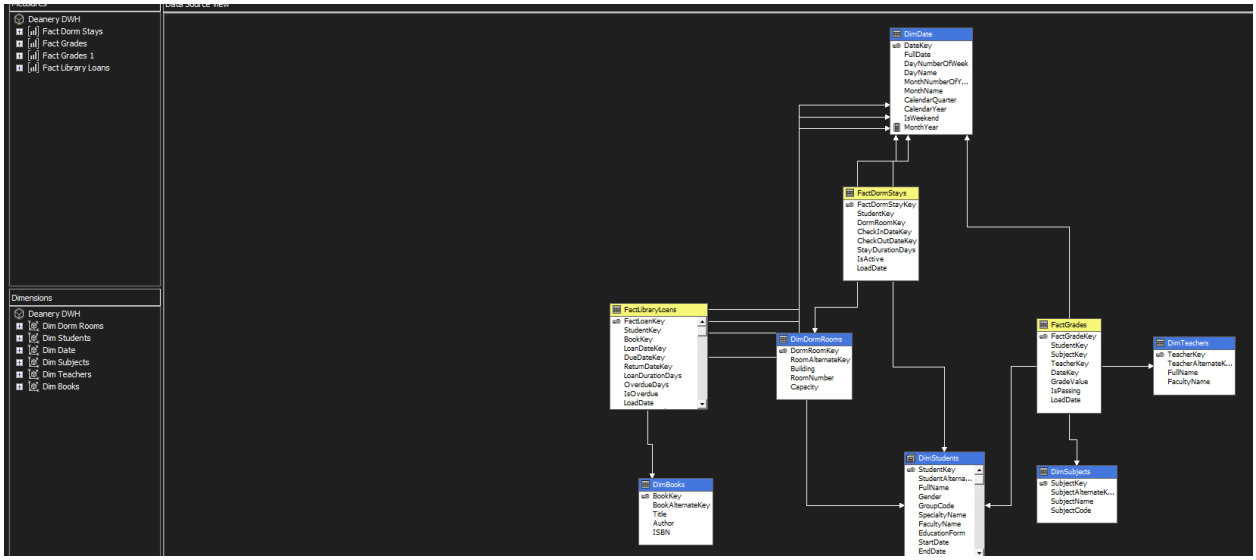


Рис 4.1.1 Початок налаштування OLAP-куба

До DSV я додав необхідний набір таблиць: 3 таблиці фактів (FactGrades, FactLibraryLoans, FactDormStays) та 6 таблиць вимірів (DimStudents, DimDate, DimSubjects, DimTeachers, DimBooks, DimDormRooms).

Службові таблиці (такі як Sys_AuditLog та системні діаграми) були виключені зі схеми, щоб не перевантажувати модель.

Під час створення DSV я пересвідчився, що система автоматично розпізнала та імпортувала зв'язки (Primary Keys / Foreign Keys), закладені в базі даних.

4.2. Проектування Data Source View

Додавання таблиць фактів та вимірів

У редакторі Data Source View я сформував повну схему даних, необхідну для аналізу.

До робочої області (Diagram) були додані три ключові таблиці фактів: FactGrades (успішність), FactLibraryLoans (бібліотека) та FactDormStays (гуртожитки).

Навколо них я розмістив таблиці вимірів: DimStudents, DimDate, DimSubjects, DimTeachers, DimBooks та DimDormRooms. Це дозволило візуалізувати архітектуру "Зірка" безпосередньо в середовищі розробки.

Визначення зв'язків між таблицями

Оскільки на рівні бази даних я попередньо створив обмеження зовнішніх ключів (Foreign Keys), SSAS автоматично розпізнав та побудував зв'язки між таблицями фактів та вимірів у DSV. Я перевірив коректність цих зв'язків на діаграмі, переконавшись, що кожна таблиця фактів коректно посилається на відповідні.

Створення іменованих обчислень (Named Calculations)

Для покращення відображення даних у звітах, не змінюючи фізичну структуру бази даних, я використав механізм іменованих обчислень.

У таблиці DimDate я створив нове обчислюване поле MonthYear.

LEFT(MonthName, 3) + ' ' + CAST(CalendarYear AS CHAR(4))

Це поле дозволяє виводити на графіках зручні підписи формату "Jan 2024" або "Sep 2025" замість стандартних дат чи номерів місяців, що значно покращує читабельність часових рядів.

Створення іменованих запитів (Named Queries)

Для розширення можливостей аналізу студентів я замінив пряме звернення до таблиці DimStudents на іменований SQL-запит (Named Query).

Я модифікував вибірку даних, додавши конкатенацію полів:

(FullName + ' (' + ISNULL(GroupCode, 'No Group') + ')') AS StudentWithGroup.

Це створило комбінований атрибут, який одразу показує ім'я студента разом із його групою (наприклад, "Іванов Іван (КН-401)"). Це спрощує

навігацію по ієрархії у майбутньому кубі, дозволяючи однозначно ідентифікувати студентів з однаковими прізвищами у різних групах.

4.3. Створення вимірів (Dimensions)

Для забезпечення багатогранного аналізу даних я розробив 6 вимірів, що перевищує мінімальні вимоги курсової роботи.

1. Часовий вимір (DimDate)

Це базовий вимір для будь-якої аналітики.

Атрибути: Я вибрав набір атрибутів, необхідних для звітності: Year, Quarter, MonthName, DayNumberOfWeek, а також створений раніше MonthYear.

Ієрархія: Мною була побудована стандартна календарна ієрархія: **Year -> Quarter -> Month -> Date**. Це дозволяє користувачам "провалюватися" (Drill-down) від річних звітів до щоденних показників.

Налаштування Order By: Я вирішив критичну проблему сортування місяців. За замовчуванням SSAS сортує текстові атрибути за алфавітом (April, August...). Я змінив налаштування атрибута MonthName, встановивши властивість OrderBy = Key та вказавши MonthNumberOfYear як ключ сортування. Тепер місяці у звітах відображаються у правильному хронологічному порядку (January, February...).

Name Column: Для ключового атрибута DateKey я налаштував відображення повної дати (FullDate), щоб користувач бачив зрозумілий формат "2024-01-01" замість системного числа.

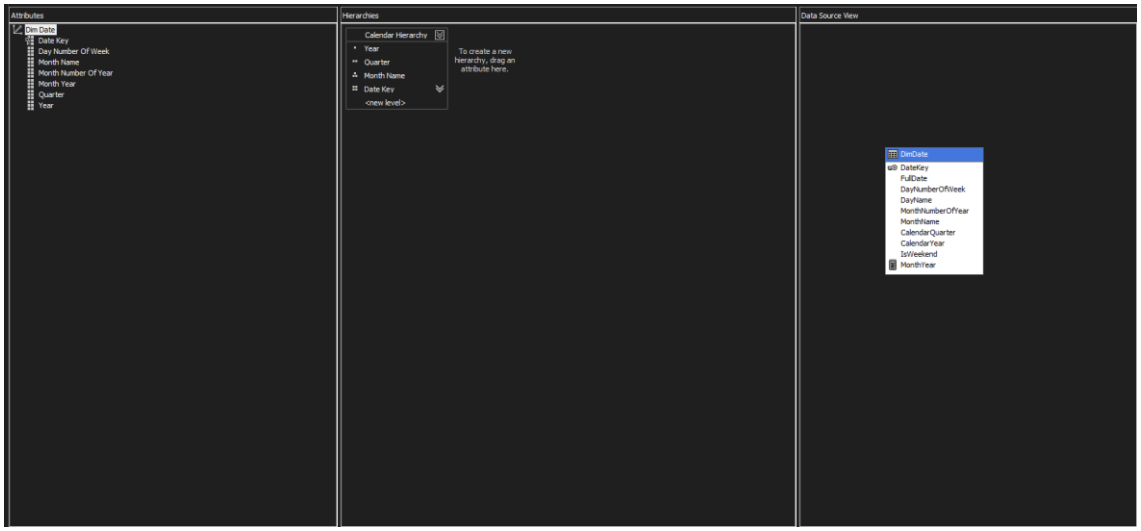


Рис 4.3.1 Створення виміру дати

2. Вимір основної сутності (DimStudents)

Цей вимір відображає головний об'єкт аналізу — студента.

Структура: Вимір базується на таблиці DimStudents, яка підтримує логіку **SCD Type 2**. Використання сурогатного ключа StudentKey дозволяє коректно аналізувати успішність студента в динаміці, навіть якщо він змінював групу чи прізвище.

Ієрархія: Я створив користувацьку ієрархію **University Structure** (Структура університету): **Faculty → Specialty → Group → Student**. Це дозволяє деканату швидко переглядати агреговані дані по факультетах і деталізувати їх до конкретної групи чи студента.

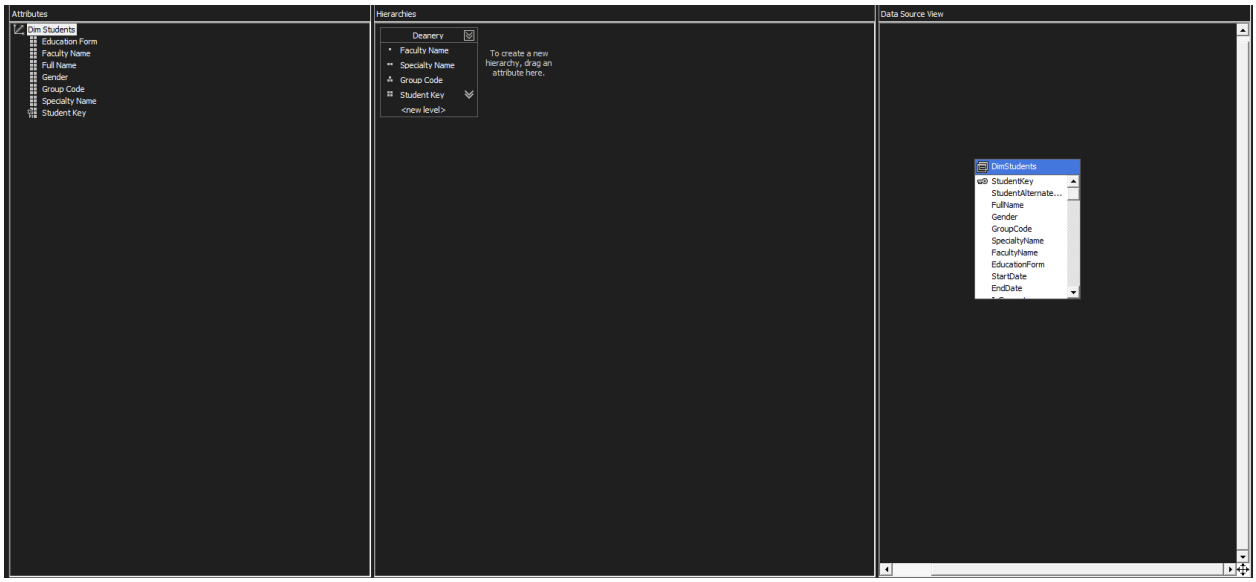


Рис 4.3.2 Створення виміру Студент

Додаткові виміри

Мною були створені специфічні для предметної області виміри:

DimSubjects: Довідник дисциплін з атрибутами назви та коду предмета.

DimTeachers: Вимір викладачів для аналізу навантаження та успішності в розрізі педагогів.

DimBooks: Вимір бібліотечного фонду (Назва, Автор, ISBN) для аналізу популярності літератури та боргів.

DimDormRooms: Вимір житлового фонду (Корпус, Номер кімнати, Місткість) для аналізу поселення.

Вимоги до вимірів

Для всіх вимірів я визначив Key Attributes (ключові атрибути), що базуються на сурогатних ключах DWH. Після налаштування я виконав операцію Process Dimension, яка успішно створила фізичну структуру вимірів на сервері SSAS.

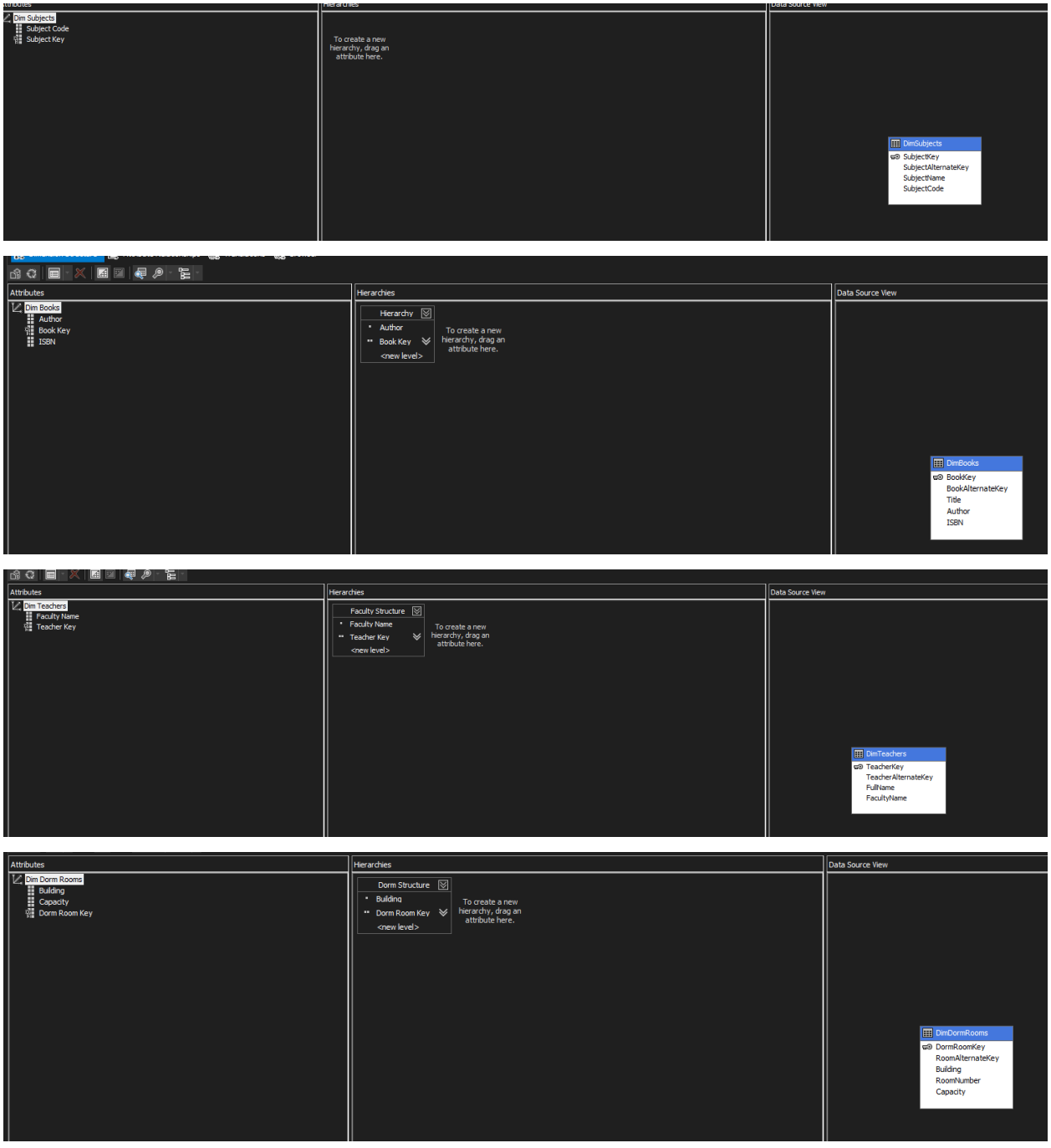


Рис 4.3.3 Інші виміри

4.4 Створення міри (Measures) та груп мір (Measure Groups)

На етапі проектування куба в середовищі SQL Server Data Tools (SSDT) я створив три групи мір, що відповідають моїм таблицям фактів у сховищі

даних: FactGrades (успішність), FactLibraryLoans (бібліотека) та FactDormStays (гуртожиток).

Для забезпечення всебічного аналізу даних я реалізував міри різних типів агрегації, як базові (фізичні), так і обчислювані.

1. Sum — Сумарні показники

Для отримання валових значень я використав стандартну функцію агрегації Sum.

У групі мір FactGrades я залишив міру Grade. Вона підсумовує бали студентів, що є проміжним кроком для подальшого розрахунку середнього балу.

У групі мір FactDormStays я використав міру Sum, яка підсумовує загальну кількість днів проживання студентів у гуртожитку, що дозволяє оцінити завантаженість кімнат.

У групі мір FactLibraryLoans я створив специфічну міру Sum (борг більше року). Оскільки це складна бізнес-логіка, спочатку я створив іменоване обчислення (Named Calculation) у Data Source View (DSV) з використанням SQL-виразу CASE WHEN..., який повертає 1 для прострочених книг та 0 для інших. У кубі я застосував до цього поля агрегацію Sum, отримавши загальну кількість "злісних" боржників.

2. Count — Підрахунок записів

Для аналізу кількості транзакцій я використав автоматично згенеровані міри з агрегацією Count:

Count of Rows— показує загальну кількість виставлених оцінок. Цей показник необхідний як знаменник для розрахунку середнього арифметичного.

Count of Rows — відображає кількість операцій видачі книг у бібліотеці.

Count of Rows — фіксує кількість фактів поселення студентів у гуртожиток.

3. Min/Max — Екстремальні значення

Для визначення пікових показників успішності я створив міру з агрегацією Maximum.

У вкладці Cube Structure для таблиці фактів FactGrades я створив нову міру Max Grade. В якості джерела даних (Source Column) я вказав стовпець Grade, а тип агрегації (Usage) встановив у Maximum. Це дозволяє миттєво знаходити найвищу оцінку в розрізі будь-якої групи, предмету чи факультету.

4. Average — Середні значення

Оскільки стандартна агрегація Average може давати похибку при роботі з порожніми значеннями (NULL) або складними ієрархіями, я реалізував розрахунок середнього значення через Calculated Member (обчислювану міру) за допомогою мови MDX.

Average Grade — розраховується за формулою ділення суми балів на їх кількість з перевіркою на ділення на нуль: [Measures].[Grade] / [Measures].[Fact Grades Count]. Це забезпечує коректне відображення рейтингу (середнього балу) студента.

5. Distinct Count — Кількість унікальних значень

Для того, щоб розрізняти кількість оцінок і реальну кількість студентів (оскільки один студент має багато оцінок), я створив міру з агрегацією Distinct Count.

У групі мір FactGrades я додав міру Unique Students Count.

У налаштуваннях я обрав Source Column: StudentKey та встановив Usage: Distinct count. Це дозволяє отримати точну кількість унікальних осіб, які навчаються або отримували оцінки у вибраний період.

6. Calculated Measures — Обчислювані міри

Використовуючи мову запитів MDX (Multidimensional Expressions), я реалізував складну бізнес-логіку, якої вимагало завдання:

Бізнес-метрика (Статус стипендії): Міра Scholarship Eligibility. Я написав MDX-вираз, який аналізує середній бал студента (Average Grade). Якщо бал перевищує встановлений поріг 73 бали, міра повертає "1" (стипендія призначена), інакше "0". Це дозволяє автоматизувати формування списків на стипендію.

Time Intelligence (Часовий ряд): Міра Loans YTD (Year To Date). Для аналізу бібліотечної активності я використав функцію YTD, яка накопичує кількість виданих книг з початку поточного року до поточної дати. Це дозволяє бачити динаміку зростання читацької активності протягом року.

Ranking (Рейтингування): Міри Student Rank by Group та Student Rank by Speciality. Я використав функцію RANK у поєднанні з ORDER, щоб присвоїти кожному студенту порядковий номер залежно від його середнього балу (Average Grade). Це дозволяє автоматично будувати рейтингові списки студентів від найкращого до найгіршого в межах групи або спеціальності.

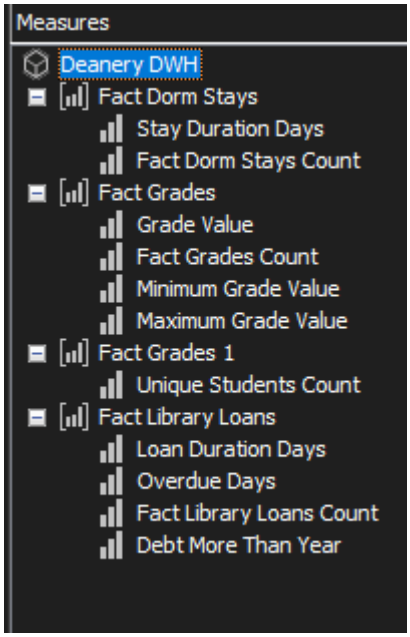


Рис 4.5.1 Створення мір

4.6 Налаштування та оптимізація куба

Для забезпечення швидкої реакції системи на запити користувачів та ефективної обробки великих масивів даних (понад 500 000 записів оцінок) я виконав комплекс дій з оптимізації фізичної структури куба.

1. Створення агрегацій (Aggregations Design)

Агрегації — це попередньо розраховані підсумкові дані, які зберігаються в кубі для прискорення виконання запитів.

Для найбільшої таблиці фактів FactGrades я використав майстер Aggregation Design Wizard.

В процесі налаштування я вказав орієнтовну кількість рядків (Row Count) — 500 000.

Як критерій оптимізації я обрав опцію Performance Gain (Приріст продуктивності) і встановив цільове значення 30%.

Система автоматично проаналізувала структуру вимірів та згенерувала набір агрегацій, які покривають найбільш ймовірні запити (наприклад, суми оцінок по факультетах або роках), що дозволило значно зменшити час відгуку звітів.

2. Створення партицій (Partitions)

Для оптимізації керування даними та прискорення їх обробки я розділив таблицю фактів FactGrades на логічні частини (партиції). Замість однієї монолітної таблиці я створив дві окремі секції, використовуючи SQL-запити в налаштуваннях джерела (Query Binding):

FactGrades_Archive: містить історичні дані за минулі роки (архів), відфільтровані за умовою WHERE [DateKey] < 20230101. Ця партиція рідко оновлюється.

FactGrades_Current: містить оперативні дані починаючи з 2023 року (WHERE [DateKey] >= 20230101). Такий підхід дозволяє пришвидшити обробку куба, оскільки при оновленні даних можна перераховувати лише актуальну партицію, не чіпаючи архів.

3. Налаштування Storage Mode (MOLAP)

Я перевірив та налаштував режим збереження даних для всіх партицій куба. Вибрано режим MOLAP (Multidimensional OLAP).

У цьому режимі детальні дані та агрегації зберігаються безпосередньо на сервері SSAS у стиснутому та оптимізованому багатовимірному форматі.

Це забезпечує найвищу швидкість виконання запитів, оскільки серверу не потрібно щоразу звертатися до реляційної бази даних (SQL Server), як у випадку з ROLAP.

4. Оптимізація атрибутних зв'язків

Під час проектування вимірів я налаштував жорсткі ієрархічні зв'язки між атрибутами (Attribute Relationships).

У вимірі DimStudents я вибудував ланцюжок залежностей: Faculty → Specialty → Group → Student.

У вимірі DimDate налаштовано зв'язок: Year → Month → Date. Це дозволяє серверу SSAS будувати індекси та агрегації, знаючи, що один студент належить лише одній групі, а група — одній спеціальності. Така оптимізація усуває необхідність у зайвих операціях з'єднання (joins) під час запиту, що суттєво підвищує продуктивність.

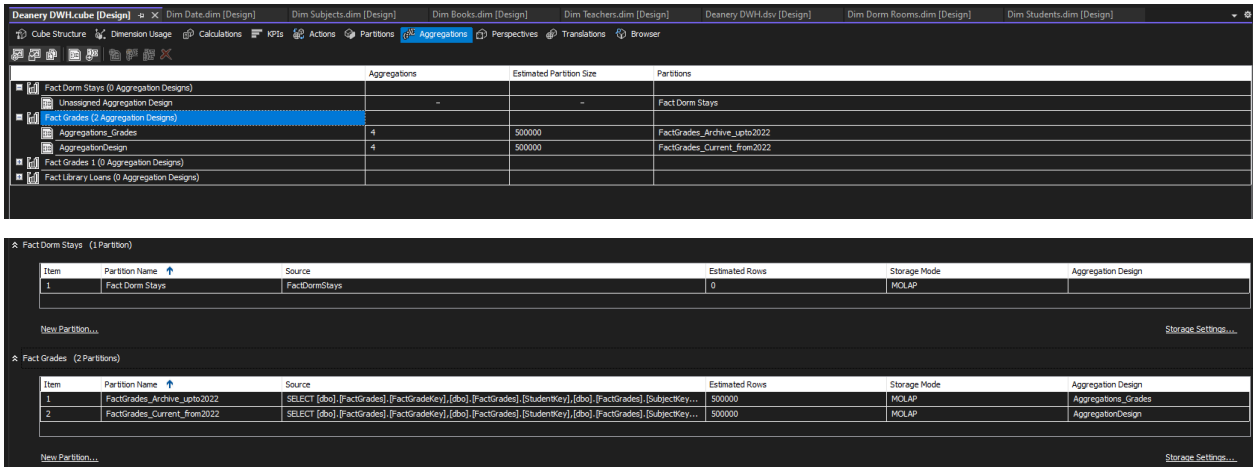


Рис 4.6.1 Оптимізація Куба

4.7 Розгортання та тестування куба

Фінальним етапом розробки OLAP-системи стало її розгортання на сервері та перевірка працездатності.

1. Розгортання куба на сервері SSAS

Перед початком розгортання я налаштував властивості проекту у Visual Studio. У вкладці Deployment я вказав цільовий сервер та назву бази даних (Database: **DeaneryDS**). Процес розгортання (Deployment) я ініціював через Solution Explorer. Середовище розробки перевірило синтаксис скриптів, XML-структуру вимірів та куба, після чого відправило метадані на сервер Analysis Services. Процес завершився успішно, про що свідчив статус "Deployment Completed Successfully" у вікні Deployment Progress.

2. Обробка куба (Process Full)

Під час первинного розгортання та після налаштування партицій система автоматично виконала операцію Process Full.

Сервер SSAS зчитав дані з реляційного сховища (SQL Server).

Дані були розподілені по створених партиціях (FactGrades_Archive та FactGrades_Current).

Були розраховані та збережені агрегації, індекси та ієрархії атрибутів. Це дозволило перетворити "порожню" структуру куба на наповнену даними аналітичну модель.

3. Перевірка коректності даних

Для верифікації даних я використав вбудований інструмент Cube Browser у Visual Studio.

Я виконав підключення до розгорнутого куба (Reconnect).

Шляхом перетягування (Drag-and-Drop) мір та вимірів я побудував тестові зрізи: перевірив відображення середнього балу по факультетах, коректність роботи ієрархії дат (Рік-Місяць) та розрахунок міри "Боржники".

Перевірка підтвердила, що всі зв'язки між таблицями фактів та вимірами працюють коректно (дані не дублюються, фільтрація відбувається правильно).

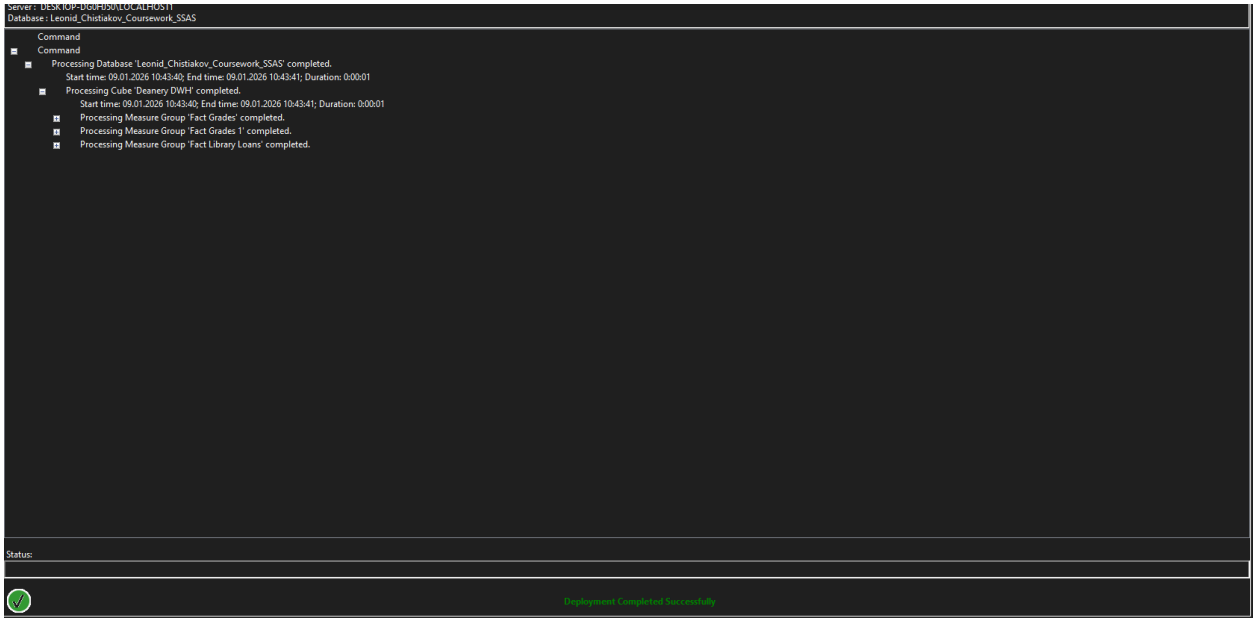


Рис 4.7.1 Деплой Куба

4. Тестування продуктивності запитів

Завдяки попередньо налаштованим агрегаціям та режиму MOLAP, час відгуку на запити у браузері був миттєвим навіть при виборі даних за весь період (5 років). Я переконався, що обчислювані міри (такі як Ranking та YTD) розраховуються коректно і не викликають затримок інтерфейсу.

5. Створення резервних копій

В якості основного методу резервного копіювання я використав збереження повного коду рішення (Solution) у Visual Studio, що включає всі визначення джерел даних (DSV), вимірів та структури куба. Це дозволяє розгорнути проект на будь-якому іншому сервері з нуля за кілька хвилин.]

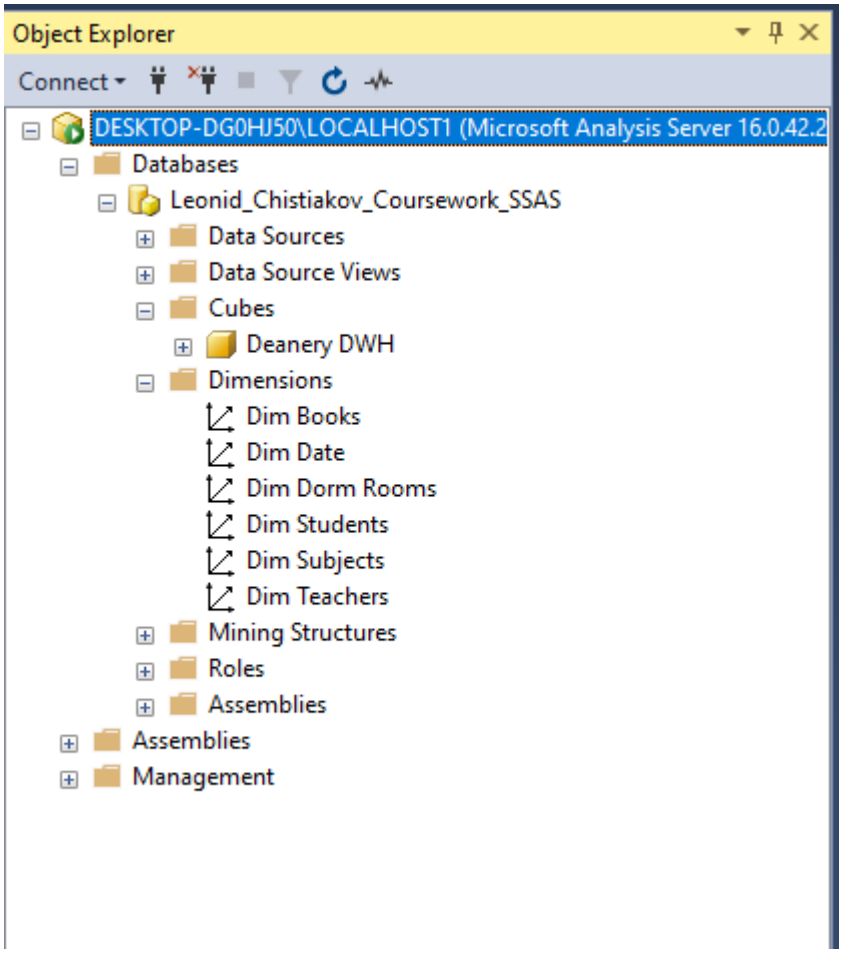


Рис 4.7.2 Куб у SSAS

Етап 5. Створення аналітичних звітів (SQL Server Reporting Services)

5.1 Створення проекту SSRS

Роботу над п'ятим етапом я розпочав із розгортання середовища для розробки звітності на базі SQL Server Reporting Services (SSRS) у середовищі Visual Studio 2022.

Створення нового проекту Report Server Project Для початку роботи я запустив Visual Studio та обрав створення нового проекту. У списку доступних шаблонів я знайшов та вибрав "Report Server Project" (Проект сервера звітів). Я свідомо відмовився від використання "Report Server Project Wizard", оскільки мені потрібен був повний контроль над структурою звітів та ручне налаштування джерел даних. Після натискання кнопки "Create" переді мною відкрився Solution Explorer з порожніми папками Shared Data Sources, Shared Datasets та Reports.

Створення Shared Data Source (Спільного джерела даних)

Наступним кроком я налаштував підключення до мого OLAP-куба. Оскільки згідно з технічним завданням мені необхідно розробити мінімум 5 різних звітів, я вирішив створити Shared Data Source (Спільне джерело даних). Це дозволяє налаштувати рядок підключення лише один раз і використовувати його у всіх звітах проекту, що значно спрощує подальшу підтримку та розгортання.

Процес налаштування виглядав так:

У вікні Solution Explorer я натиснув правою кнопкою миші на папку Shared Data Sources та обрав пункт Add New Data Source.

У діалоговому вікні, що відкрилося, я задав ім'я джерела — DeaneryDS.

У випадаючому списку Type (Тип) я змінив стандартне значення на Microsoft SQL Server Analysis Services. Це критично важливий момент, оскільки ми підключаємося не до реляційної бази даних, а до багатовимірного куба SSAS.

Налаштування підключення до OLAP-куба

Далі я натиснув кнопку Edit для формування рядка підключення (Connection String):

У полі Server name я вказав ім'я свого локального сервера (або localhost), де розгорнуто службу Analysis Services.

У блоці Connect to a database розгорнув список доступних баз даних.

Я обрав свою базу даних Deanery, яку я задеплоїв на попередньому етапі роботи з SSAS. Саме в ній міститься мій куб DeaneryCube.

Для перевірки коректності налаштувань я натиснув кнопку Test Connection. Системне повідомлення "Test connection succeeded" підтвердило, що Visual Studio успішно бачить мій куб.

Після натискання ОК я завершив налаштування. Тепер у папці Shared Data Sources з'явився файл DeaneryDS.rds, який став фундаментом для всіх подальших звітів.

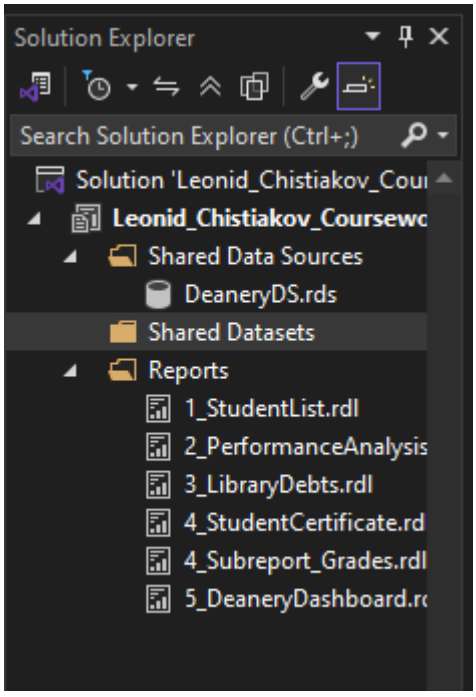


Рис 5.1.1 Підключення бази даних

5.2 Розробка звітів

На цьому етапі я перейшов до безпосереднього створення звітів, кожен з яких демонструє різні можливості SSRS та відповідає специфічним вимогам курсової роботи.

Табличний звіт (Table Report)

Першим звітом, який я реалізував, став "Список студентів по групах". Цей звіт призначений для перегляду детальної інформації про успішність студентів з можливістю фільтрації.

Підготовка даних та параметрів

Роботу я розпочав зі створення наборів даних (Datasets). Оскільки структура університету ієрархічна, я реалізував механізм каскадних параметрів (Cascading Parameters), щоб користувачу було зручно обирати потрібну групу:

Спочатку я створив Dataset Faculties, який витягує унікальний список факультетів з куба. На його основі я створив параметр @Faculty.

Далі створив Dataset Specialties, який залежить від обраного факультету (використовуючи MDX-функцію STRTOMEMBER), та відповідний параметр @Specialty.

Аналогічно створив Dataset Groups, залежний від спеціальності.

Головний набір даних MainData я налаштував так, щоб він отримував список студентів, їхній середній бал, рейтинг та статус стипендії, фільтруючись за обраним параметром групи.

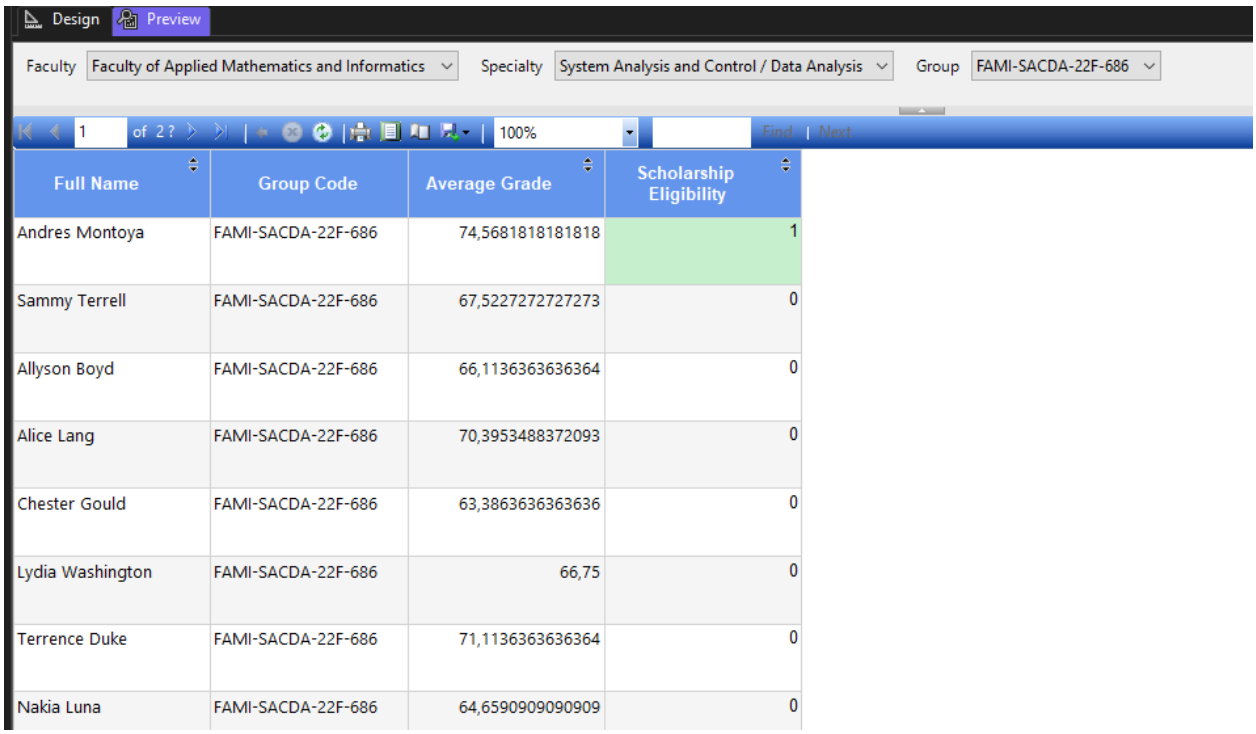
Візуалізація та функціонал

Для відображення даних я використав елемент Table.

Сортування: Для стовпця "Середній бал" я увімкнув інтерактивне сортування (Interactive Sorting), що дозволяє користувачу в режимі перегляду одним кліком впорядкувати студентів від відмінників до двієчників.

Умовне форматування (Conditional Formatting): Щоб візуально виділити успішність, я прописав вираз для властивості BackgroundColor. Комірки з оцінкою нижче 60 балів автоматично забарвлюються у червоний колір, а вище 90 — у зелений.

Оформлення: Для покращення читабельності великих списків я застосував ефект "зебри" (чергування кольорів рядків) за допомогою функції RowNumber.



Full Name	Group Code	Average Grade	Scholarship Eligibility
Andres Montoya	FAMI-SACDA-22F-686	74,5681818181818	1
Sammy Terrell	FAMI-SACDA-22F-686	67,5227272727273	0
Allyson Boyd	FAMI-SACDA-22F-686	66,1136363636364	0
Alice Lang	FAMI-SACDA-22F-686	70,3953488372093	0
Chester Gould	FAMI-SACDA-22F-686	63,3863636363636	0
Lydia Washington	FAMI-SACDA-22F-686	66,75	0
Terrence Duke	FAMI-SACDA-22F-686	71,1136363636364	0
Nakia Luna	FAMI-SACDA-22F-686	64,6590909090909	0

Рис 5.2.1 Звіт-таблиця

Матричний звіт (Matrix Report)

Для аналізу успішності я розробив складний матричний звіт, який дозволяє аналізувати дані у двох розрізах: по викладачах та по предметах.

Динамічна структура

Я вирішив не обмежуватися статичною матрицею і реалізував динамічний вибір осей аналізу.

Я створив параметр @AnalysisMode з двома значеннями: "По викладачах" та "По предметах".

Додав параметр @Teachers та @Subjects з можливістю множинного вибору (Multi-select), щоб користувач міг порівнювати конкретних викладачів або предмети.

Налаштування Matrix

Я використав елемент Matrix (Cross-tab). Замість жорсткої прив'язки полів до рядків та колонок, я використав вирази (Expressions):

Якщо обрано режим "По викладачах", у рядках (Row Groups) виводяться прізвища викладачів, а в колонках (Column Groups) — назви предметів.

Якщо режим перемикається на "По предметах", матриця автоматично "перевертається" (Pivot), міняючи місцями рядки та колонки.

На перетині (у Data Cells) я розмістив міру [Average Grade], яка автоматично агрегується кубом для обраного перетину викладача та предмета. Також я додав стовпчикову діаграму (Column Chart) під матрицею, яка візуалізує ці дані, динамічно змінюючи підписи осі X залежно від обраного режиму.

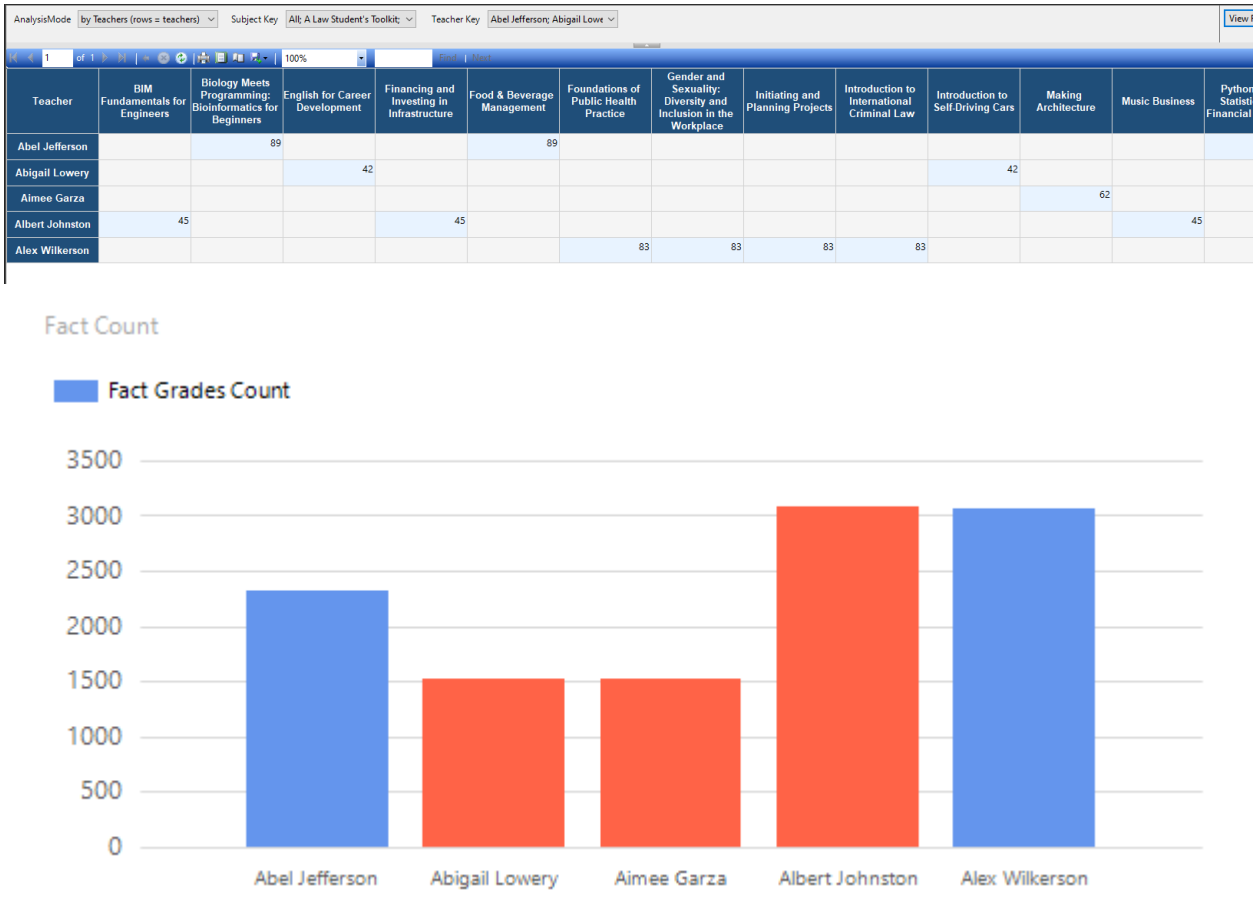


Рис 5.2.2 Матричний звіт

Деталізований звіт з drill-down

Для виконання вимоги щодо звітності по заборгованостях у бібліотеці я створив звіт "Заборгованість по книгах" (3_LibraryDebts), використовуючи технологію Drill-down.

Групування даних

У цьому звіті я використав таблицю з трьома рівнями вкладеності:

Рівень 1 (Факультет): Верхній рівень групування.

Рівень 2 (Спеціальність): Вкладена група всередині факультету.

Рівень 3 (Група): Вкладена група всередині спеціальності.

Рівень 2 (Студент): Вкладена група всередині групи.

Деталі (Details): Конкретні книги, та статус боргу та тривалість боргу.

Реалізація Drill-down

Щоб звіт не виглядав громіздким при завантаженні, я налаштував властивість Visibility:

Рядки спеціальностей, груп, студентів приховані за замовчуванням і відкриваються лише при натисканні на значок "плюс" біля назви факультету (Toggle Item).

Деталі по книгах приховані, доки користувач не розгорне конкретного студента.

Параметризація

Додатково я створив Boolean-параметр @ShowOnlyHardDebtors ("Показати тільки боржників > 1 року"). Я використав його для фільтрації групи Details, використовуючи створену в кубі міру Debt More Than Year. Це дозволяє деканату миттєво відфільтрувати список і побачити лише критичні заборгованості, приховавши тих, хто повернув книги вчасно.

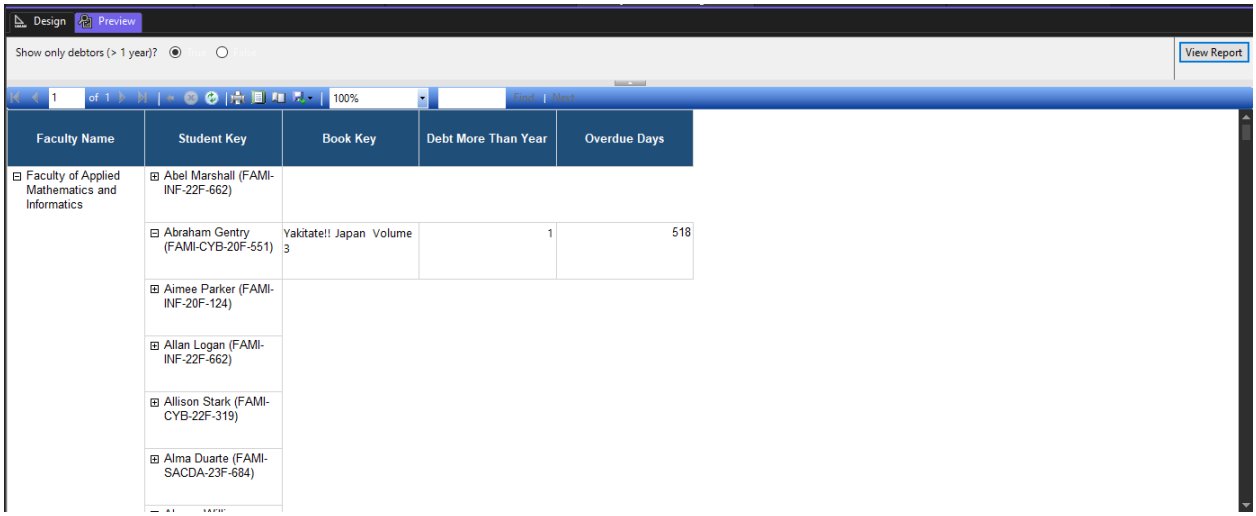


Рис 5.2.3 Drill-down звіт

5.3 Параметризація звітів

Для забезпечення інтерактивності та зручності користування звітами я реалізував чотири типи параметрів, кожен з яких вирішує конкретну бізнес-задачу фільтрації даних.

1. Каскадні параметри (Cascading Parameters)

Найскладнішу логіку залежних списків я реалізував у звіті "Список студентів по групах". Оскільки вибирати групу із загального списку (де їх сотні) незручно, я створив ланцюжок залежностей: Факультет → Спеціальність → Група.

Спочатку я створив параметр @Faculty, значення якого підтягуються з Dataset Faculties.

Для параметра @Specialty я створив окремий Dataset, у якому використав значення обраного факультету для фільтрації запиту. Це дозволило динамічно оновлювати список спеціальностей при зміні факультету.

Аналогічно налаштував параметр @Group, який залежить від обраної спеціальності. Тільки після вибору всіх трьох рівнів формується фінальний звіт.

2. Multi-select параметри (Множинний вибір)

У матричному звіті "Аналіз успішності" я надав користувачу можливість порівнювати між собою довільну кількість викладачів або предметів.

У налаштуваннях параметрів @Teachers та @Subjects я активував опцію Allow multiple values.

Це автоматично додало функціонал "Select All" (Обрати всіх) у випадаючий список.

SSRS автоматично передає обрані значення як масив у запит до OLAP-куба, що дозволяє будувати матрицю лише для обраного піднабору даних.

3. Boolean параметри (Перемикачі)

Для звіту "Заборгованість по книгах" я використав логічний параметр для швидкої фільтрації критичних даних.

Я створив параметр @ShowOnlyHardDebtors з типом даних Boolean.

Як значення за замовчуванням встановив True.

У властивостях таблиці (Tablix Properties -> Filters) я прописав вираз, який перевіряє значення цього параметра: якщо він істинний, таблиця показує лише рядки, де міра [Debt More Than Year] більше нуля. Це дозволяє деканату одним кліком приховувати "сумлінних" студентів і фокусуватися на боржниках.

4. Dropdown параметри (Випадаючі списки)

Для формування "Довідки студента" мені потрібно було забезпечити точний вибір конкретної особи.

Я створив параметр @StudentKey, який використовує Dataset зі списком усіх студентів.

Важливою деталлю стало налаштування полів: для відображення користувачу (Label Field) я обрав поле Student With Group (Ім'я + Група), а для передачі в запит (Value Field) — унікальний ключ StudentKey. Це дозволяє користувачу шукати студента за зрозумілим ім'ям, а звіту — працювати з точним ідентифікатором.

5.4 Додаткова функціональність звітів

Для підвищення зручності користування та інформативності розроблених звітів я впровадив низку додаткових функцій, що дозволяють візуально виділяти важливі дані, взаємодіяти з таблицями та формувати друковані документи.

1. Conditional Formatting (Умовне форматування)

Щоб користувач міг миттєво оцінити ситуацію з успішністю, я налаштував динамічну зміну кольору комірок залежно від значень даних.

У звітах "Список студентів" та "Аналіз успішності" я використав вирази (Expressions) для властивості BackgroundColor.

За допомогою функції Switch я реалізував логіку: якщо середній бал студента нижчий за 60, комірка підсвічується червоним кольором, а якщо вищий за 90 — зеленим.

Додатково я реалізував ефект "зебри" (чергування кольорів рядків), використавши функцію RowNumber та оператор залишку від ділення (Mod). Це значно покращило читабельність довгих табличних списків.

2. Interactive Features (Інтерактивні можливості)

Я зробив звіти не статичними картинками, а інструментами для аналізу:

Sorting (Інтерактивне сортування): У табличному звіті я увімкнув функцію Interactive Sorting для заголовків колонок. Це дозволило користувачу в режимі перегляду сортувати список студентів за прізвищем або за рейтингом одним кліком миші, без необхідності перезавантажувати звіт із новими параметрами.

Toggle Items (Розгортання/згортання): У звіті "Заборгованість по книгах" я використав механізм Drill-down. Я налаштував властивість Visibility для груп рядків так, щоб детальні дані (список книг) були приховані, доки користувач не натисне на "плюс" біля прізвища студента. Це дозволило компактно відобразити великий обсяг ієрархічних даних (Факультет → Спеціальність → Група → Студент → Книга).

3. Subreports (Вкладені звіти)

Для реалізації складної форми "Довідка студента" я використав технологію вкладених звітів.

Я розробив окремий допоміжний звіт, що містить лише таблицю з оцінками.

Цей звіт я вбудував у головний звіт-бланк за допомогою елемента Subreport.

Зв'язок між звітами я налаштував через передачу параметра @StudentKey: головний звіт передає ID обраного студента у вкладений звіт, який повертає відповідний перелік предметів та оцінок. Це дозволило створити документ зі складною верстkою, де поєднуються вільний текст шапки та структурована таблиця.

4. Export Functionality (Функціональність експорту)

Оскільки курсова робота передбачає створення офіційних документів (довідок), я протестував та оптимізував відображення звітів для експорту.

Зокрема, для звіту "Довідка студента" я налаштував розміри полів та розміщення елементів так, щоб при експорті у PDF документ виглядав як готовий до друку бланк формату A4, без зайвих розривів сторінок та зміщення меж таблиць. Перевірка показала коректне збереження форматування при вивантаженні файлу.

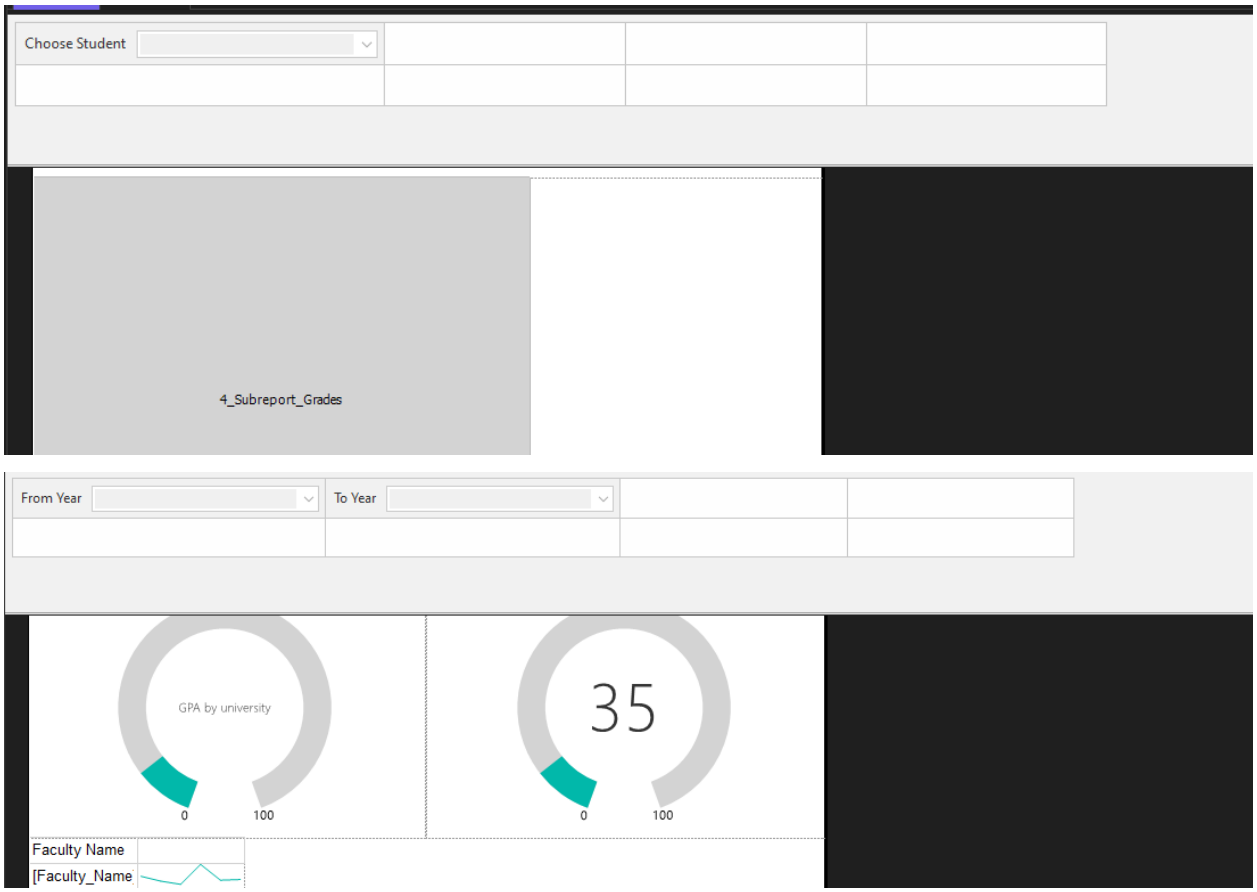


Рис 5.4.1 Інші звіти

5.5 Розгортання звітів

Фінальним етапом роботи стало розгортання розробленого рішення на сервері звітів (SQL Server Reporting Services) для забезпечення доступу кінцевих користувачів (співробітників деканату) до аналітики через веб-інтерфейс.

Публікація звітів на Report Server Процес розгортання я виконав безпосередньо із середовища Visual Studio.

У властивостях проекту (Project Properties) я налаштував конфігурацію TargetServerURL, вказавши адресу мого локального сервера звітів.

Також я визначив параметр TargetReportFolder, задавши назву кореневої папки — Deanery_Analytics.

Розгортання я проводив у два етапи: спочатку опублікував спільне джерело даних (DeaneryDS), щоб гарантувати наявність підключення на сервері, а потім виконав команду Deploy для всього проекту. У вікні виводу я отримав підтвердження про успішне завантаження всіх файлів .rdl на сервер.

Створення папок для організації звітів Для впорядкування контенту на веб-порталі SSRS я організував структуру каталогів.

У кореневій папці Deanery_Analytics я розмістив основні аналітичні звіти.

Для службових компонентів, таких як вкладені звіти (зокрема 4_Subreport_Grades), я створив приховану підпапку System_Subreports. Це дозволило прибрати технічні файли з головного екрана користувача, залишивши лише фінальні документи, при цьому не порушуючи функціональність виклику підзвітів.

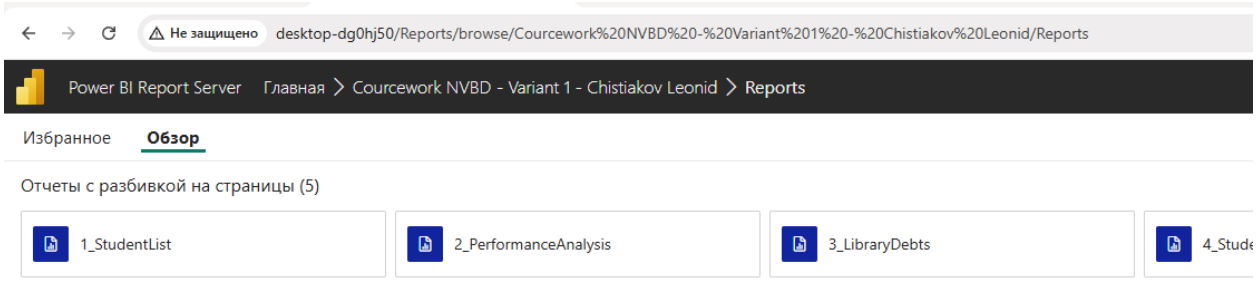


Рис 5.5.1 Звіти у Power BI

Налаштування безпеки та доступу

Після публікації я перейшов до веб-порталу для налаштування прав доступу. Використовуючи рольову модель безпеки SSRS:

Собі, як адміністратору системи, я залишив роль Content Manager, що дає повний контроль над об'єктами.

Для групи користувачів "Співробітники Деканату" я налаштував роль Browser. Це дозволяє їм переглядати звіти, використовувати параметри та

експортувати дані, але забороняє змінювати структуру звітів або налаштування джерел даних.

Налаштування кешування звітів

Оскільки деякі звіти (наприклад, "Матриця успішності") звертаються до великого обсягу даних у кубі і можуть формуватися кілька секунд, я налаштував політику кешування.

Для найбільш "важких" звітів я увімкнув опцію Cache a temporary copy of the report.

Я встановив час життя кешу у 30 хвилин. Це означає, що якщо кілька співробітників відкриють один і той самий звіт протягом цього часу, сервер не буде робити повторний запит до бази даних, а миттєво віддасть збережену копію, що значно знижує навантаження на систему.

Створення Subscriptions (розсилка звітів)

Для автоматизації роботи деканату я налаштував підписку на звіт "Заборгованість по книгах".

Я створив Standard Subscription із розкладом виконання: щопонеділка о 09:00 ранку.

Формат доставки: Windows File Share. Звіт автоматично генерується у форматі Excel та зберігається у спільну мережеву папку деканату.

Це дозволяє відповідальним особам отримувати актуальний список боржників на початку робочого тижня без необхідності заходити на портал і формувати звіт вручну.

Висновок

У ході виконання курсової роботи мною було спроектовано та реалізовано комплексну інформаційно-аналітичну систему для предметної області "Деканат". Головною метою роботи було створення рішення, здатного ефективно опрацьовувати великі масиви даних (понад 10 000 студентів та 500 000 записів успішності) та надавати інструменти для оперативного прийняття управлінських рішень. Проект охопив повний життєвий цикл розробки корпоративного сховища даних: від проектування реляційної схеми до візуалізації бізнес-показників.

На етапі побудови сховища даних (DWH) було успішно реалізовано ETL-процеси за допомогою SQL Server Integration Services (SSIS). Мені вдалося забезпечити коректну екстракцію, очищення та завантаження даних із транзакційної системи в аналітичну структуру типу "Зірка". Особливу увагу було приділено підтримці історичності даних: впровадження механізму Slowly Changing Dimensions (SCD Type 1) для виміру "Студенти" дозволило коректно відстежувати зміни даних студентів протягом 5 років навчання.

Ключовим елементом системи став багатовимірний OLAP-куб, розроблений у середовищі SQL Server Analysis Services (SSAS). Створена модель даних об'єднала розрізнені бізнес-процеси (успішність, бібліотека, гуртожиток) у єдиний аналітичний простір. Завдяки використанню мови MDX було реалізовано складну бізнес-логіку: розрахунок середньозважених оцінок, динамічне формування рейтингів студентів та виявлення довгострокових заборгованостей. Застосування партиціювання таблиць фактів та налаштування агрегацій у режимі MOLAP дозволило досягти високої швидкодії запитів навіть при значних обсягах даних.

Фінальним етапом стала розробка системи звітності на базі SQL Server Reporting Services (SSRS). Реалізований набір звітів покриває всі потреби користувачів: від формування офіційних довідок із суворою структурою до інтерактивних аналітичних панелей (Dashboard) з використанням Drill-down навігації та візуалізації KPI. Впровадження параметризації, автоматичної

розсилки звітів та розмежування прав доступу перетворило розроблену систему на повноцінний інструмент для автоматизації роботи деканату.

Таким чином, у результаті виконання курсової роботи я набув практичних навичок роботи з технологічним стеком Microsoft BI, засвоїв принципи проектування надвеликих баз даних та створив працездатну систему, яка вирішує актуальні задачі моніторингу навчального процесу та адміністративно-господарської діяльності університету.

Бібліографічний список

Основна література:

1. Microsoft SQL Server Integration Services (SSIS) Documentation
2. Microsoft SQL Server Analysis Services (SSAS) Documentation
3. Microsoft SQL Server Reporting Services (SSRS) Documentation
4. "The Data Warehouse Toolkit" - Ralph Kimball
5. "Microsoft SQL Server 2019: A Beginner's Guide" - Dusan Petkovic

Додаткові ресурси:

• Microsoft Learn (офіційна документація)
• SQL Server Central
• SQLBI (для Analysis Services)
• Stack Overflow

Інструменти:

• SQL Server Management Studio (SSMS) - для роботи з БД
• SQL Server Data Tools (SSDT) - для SSIS, SSAS, SSRS
• Redgate SQL Data Generator - для генерації даних
• Visual Studio - середовище розробки
• Power BI Desktop

Додатки

Створення DeaneryDWH

```
USE master;
GO

IF NOT EXISTS (SELECT name FROM sys.databases WHERE name = N'DeaneryDWH')
BEGIN
    CREATE DATABASE [DeaneryDWH];
END
GO

USE [DeaneryDWH];
GO

IF OBJECT_ID('dbo.FactGrades', 'U') IS NOT NULL DROP TABLE dbo.FactGrades;
IF OBJECT_ID('dbo.FactLibraryLoans', 'U') IS NOT NULL DROP TABLE dbo.FactLibraryLoans;
IF OBJECT_ID('dbo.DimStudents', 'U') IS NOT NULL DROP TABLE dbo.DimStudents;
IF OBJECT_ID('dbo.DimSubjects', 'U') IS NOT NULL DROP TABLE dbo.DimSubjects;
IF OBJECT_ID('dbo.DimTeachers', 'U') IS NOT NULL DROP TABLE dbo.DimTeachers;
IF OBJECT_ID('dbo.DimBooks', 'U') IS NOT NULL DROP TABLE dbo.DimBooks;
IF OBJECT_ID('dbo.DimDate', 'U') IS NOT NULL DROP TABLE dbo.DimDate;
GO
CREATE TABLE dbo.DimDate (
    DateKey INT PRIMARY KEY,
    FullDate DATE NOT NULL,
    DayNumberOfWeek TINYINT NOT NULL,
    DayName NVARCHAR(20) NOT NULL,
    MonthNumberOfYear TINYINT NOT NULL,
    MonthName NVARCHAR(20) NOT NULL,
    CalendarQuarter TINYINT NOT NULL,
    CalendarYear SMALLINT NOT NULL,
    IsWeekend BIT NOT NULL
);
GO

-- 3.2. DimSubjects (Предмети)
CREATE TABLE dbo.DimSubjects (
    SubjectKey INT IDENTITY(1,1) PRIMARY KEY,
    SubjectAlternateKey INT NOT NULL,
    SubjectName NVARCHAR(600) NOT NULL,
    SubjectCode NVARCHAR(40) NULL
);
GO

CREATE TABLE dbo.DimTeachers (
    TeacherKey INT IDENTITY(1,1) PRIMARY KEY,
    TeacherAlternateKey INT NOT NULL,
    FullName NVARCHAR(400) NOT NULL,
    FacultyName NVARCHAR(400) NULL
);
GO

CREATE TABLE dbo.DimBooks (
    BookKey INT IDENTITY(1,1) PRIMARY KEY,
    BookAlternateKey INT NOT NULL,
    Title NVARCHAR(500) NOT NULL,
    Author NVARCHAR(400) NULL,
    ISBN NVARCHAR(40) NULL
);
GO

CREATE TABLE dbo.DimStudents (
```

```
    StudentKey INT IDENTITY(1,1) PRIMARY KEY,
    StudentAlternateKey INT NOT NULL,
    FullName NVARCHAR(400) NOT NULL,
    Gender CHAR(1) NULL,

    GroupCode NVARCHAR(100) NULL,          --
    SpecialtyName NVARCHAR(400) NULL,      --
    FacultyName NVARCHAR(400) NULL,        --
    EducationForm NVARCHAR(40) NULL,       --

    StartDate DATETIME NOT NULL,
    EndDate DATETIME NULL,                 --
    IsCurrent BIT NOT NULL DEFAULT 1       -- ]
);
GO

CREATE TABLE dbo.FactGrades (
    FactGradeKey BIGINT IDENTITY(1,1) PRIMARY KEY,

    StudentKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimStudents(StudentKey),
    SubjectKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimSubjects(SubjectKey),
    TeacherKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimTeachers(TeacherKey),
    DateKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimDate(DateKey),
    GradeValue TINYINT NOT NULL,
    IsPassing BIT NOT NULL DEFAULT 0,      -

    LoadDate DATETIME DEFAULT GETDATE()
);
GO

CREATE TABLE dbo.FactLibraryLoans (
    FactLoanKey BIGINT IDENTITY(1,1) PRIMARY KEY,

    StudentKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimStudents(StudentKey),
    BookKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimBooks(BookKey),

    LoanDateKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimDate(DateKey),    --
    DueDateKey INT NOT NULL FOREIGN KEY REFERENCES dbo.DimDate(DateKey),     --
    ReturnDateKey INT NULL FOREIGN KEY REFERENCES dbo.DimDate(DateKey),      --
    LoanDurationDays INT NULL,             -=
    OverdueDays INT NULL,
    IsOverdue BIT NOT NULL DEFAULT 0,

    LoadDate DATETIME DEFAULT GETDATE()
);
GO
DECLARE @StartDate DATE = '2020-01-01';
DECLARE @EndDate DATE = '2025-12-31';

WHILE @StartDate <= @EndDate
BEGIN
    INSERT INTO dbo.DimDate (
        DateKey,
        FullDate,
        DayNumberOfWeek,
        DayName,
        MonthNumberOfYear,
        MonthName,
        CalendarQuarter,
        CalendarYear,
        IsWeekend
    )
    VALUES (
        CAST(CONVERT(VARCHAR(8), @StartDate, 112) AS INT), --
```

```
        @StartDate,
        DATEPART(dw, @StartDate),
        DATENAME(dw, @StartDate),
        MONTH(@StartDate),
        DATENAME(mm, @StartDate),
        DATEPART(qq, @StartDate),
        YEAR(@StartDate),
        CASE WHEN DATEPART(dw, @StartDate) IN (1, 7) THEN 1 ELSE 0 END
    );

    SET @StartDate = DATEADD(dd, 1, @StartDate);
END
GO

    PRINT 'База даних DeaneryDWH та структура таблиць успішно створені!';
```