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

Кафедра прикладної математики

Звіт
про виконання курсової роботи
з курсу «Надвеликі бази даних»

Виконав:
студент групи ПМ-42
Заставний В. Є.

Перевірила:
Любінський Б. Б.

_____
(дата) (підпис викладача)

Львів – 2026 р.

Вступ

У сучасних умовах цифровізації бізнес-процесів та стрімкого зростання обсягів інформації особливої актуальності набувають системи зберігання та аналітичної обробки надвеликих баз даних. Традиційні транзакційні бази даних (OLTP), хоча й ефективні для оперативної обробки даних, не завжди забезпечують достатню продуктивність та зручність для виконання складних аналітичних запитів, формування агрегованих показників і побудови управлінської звітності.

Для вирішення зазначених проблем широко застосовуються сховища даних (Data Warehouse) та технології багатовимірного аналізу (OLAP), які дозволяють інтегрувати дані з різних джерел, виконувати їх очищення та трансформацію, а також забезпечувати швидкий доступ до аналітичної інформації. Важливу роль у таких системах відіграють ETL-процеси (Extract, Transform, Load), OLAP-куби та інструменти візуалізації даних, що надають користувачам можливість інтерактивного аналізу та прийняття обґрунтованих управлінських рішень.

Метою даної курсової роботи є проєктування та реалізація інформаційно-аналітичної системи на основі надвеликих баз даних з використанням платформи Microsoft SQL Server. У межах роботи здійснюється повний цикл створення аналітичної системи — від проєктування реляційної бази даних і наповнення її великим обсягом даних до побудови сховища даних, OLAP-куба та набору аналітичних звітів.

Для досягнення поставленої мети у курсовій роботі необхідно вирішити такі завдання:

- проаналізувати предметну область та визначити основні бізнес-процеси;
- спроєктувати концептуальну, логічну та фізичну моделі реляційної бази даних;
- реалізувати базу даних у середовищі Microsoft SQL Server та наповнити її значним обсягом тестових даних;

- розробити ETL-процеси з використанням SQL Server Integration Services (SSIS) для перенесення даних у сховище даних;
- побудувати сховище даних з використанням багатовимірної моделі (схема «зірка» або «сніжинка»);
- створити OLAP-куб за допомогою SQL Server Analysis Services (SSAS) з реалізацією аналітичних мір та обчислень;
- розробити набір аналітичних звітів у SQL Server Reporting Services (SSRS) із використанням табличних, матричних, графічних та інтерактивних елементів;
- виконати аналіз продуктивності та коректності роботи створеної аналітичної системи.

Об’єктом дослідження є процеси зберігання та аналітичної обробки великих обсягів даних.

Предметом дослідження є методи проєктування сховищ даних, ETL-процесів, OLAP-кубів та аналітичних звітів.

Практична цінність роботи полягає у формуванні навичок проєктування та реалізації повноцінної BI-системи, що може бути використана для аналізу даних у реальних інформаційних системах.

У межах курсової роботи реалізовано повний цикл побудови інформаційно-аналітичної системи на основі надвеликих баз даних. На початковому етапі спроєктовано та реалізовано транзакційну реляційну базу даних (OLTP) у середовищі Microsoft SQL Server, яка містить нормалізовані таблиці предметної області та наповнена значним обсягом тестових даних.

На основі OLTP-бази створено сховище даних (Data Warehouse), побудоване за багатовимірною моделлю типу «зірка». Перенесення та обробка даних здійснюється за допомогою ETL-процесів, реалізованих у SQL Server Integration Services (SSIS), із використанням трансформацій для очищення,

перевірки та агрегації даних, а також механізмів обробки помилкових записів.

Для забезпечення аналітичної обробки даних побудовано OLAP-куб за допомогою SQL Server Analysis Services (SSAS), який містить набір вимірів та обчислюваних мір і підтримує багатовимірний аналіз даних.

На завершальному етапі розроблено комплекс аналітичних звітів у SQL Server Reporting Services (SSRS), зокрема табличні, матричні, графічні звіти та інтерактивну панель керування (dashboard), що забезпечують можливість фільтрації, деталізації (drill-down) та візуалізації аналітичних показників.

Опис предметної області

Предметною областю даної курсової роботи є інформаційна система обліку та аналізу діяльності навчального закладу. Система призначена для зберігання, обробки та аналітичного аналізу даних, пов’язаних з організацією навчального процесу, структурою закладу, контингентом студентів та результатами їх навчання.

У межах предметної області розглядається ієрархічна структура навчального закладу, яка включає факультети, кафедри та навчальні дисципліни. Кожна кафедра відповідає за викладання певного набору дисциплін, які, у свою чергу, закріплюються за відповідними освітніми програмами або напрямами підготовки.

Ключовим об’єктом предметної області є студент, який навчається у певному навчальному закладі, належить до конкретного факультету та кафедри, а також проходить навчання з визначеного набору дисциплін. Для кожного студента зберігаються персональні та навчальні дані, необхідні для обліку та подальшого аналізу.

У процесі навчання студенти отримують оцінки з дисциплін у межах окремих навчальних періодів (семестрів, навчальних років). Дані про результати навчання використовуються для аналізу успішності студентів, порівняння

показників між кафедрами, факультетами та часовими періодами, а також для формування зведеної аналітичної звітності.

Предметна область передбачає накопичення значних обсягів історичних даних за декілька навчальних років, що створює необхідність використання сховища даних та OLAP-технологій для ефективного аналізу. Основна увага приділяється не лише зберіганню інформації, а й можливості швидкого отримання агрегованих показників, таких як середній бал, кількість студентів, динаміка успішності та порівняльний аналіз між структурними підрозділами навчального закладу.

Таким чином, розглянута предметна область є типовим прикладом задачі аналітичної обробки надвеликих обсягів даних, де поєднуються транзакційні операції обліку та складні багатовимірні аналітичні запити, що обґрунтовує використання технологій ETL, сховищ даних, OLAP-кубів та аналітичних звітів.

Основні бізнес-процеси предметної області

Функціонування інформаційної системи обліку та аналізу діяльності навчального закладу базується на сукупності взаємопов’язаних бізнес-процесів, що забезпечують повний цикл збирання, зберігання та аналітичної обробки навчальних даних.

Одним з основних бізнес-процесів є процес формування структури навчального закладу. У межах цього процесу здійснюється створення та підтримка довідкової інформації про факультети, кафедри та навчальні дисципліни. Дані цього типу мають відносно стабільний характер, проте з часом можуть змінюватися у зв’язку з реорганізацією структури закладу або оновленням навчальних програм.

Наступним ключовим бізнес-процесом є зарахування та облік студентів. У рамках даного процесу здійснюється реєстрація студентів, їх прив’язка до відповідних факультетів, кафедр та освітніх програм. Також зберігається інформація про статус студента, період навчання та інші облікові дані, необхідні

для подальшого аналізу контингенту.

Важливу роль у предметній області відіграє бізнес-процес організації навчального процесу. Він включає формування переліку дисциплін, визначення навчальних періодів (семестрів, навчальних років), а також закріплення дисциплін за студентами або групами студентів. Цей процес забезпечує логічний зв’язок між студентами та навчальними дисциплінами.

Процес оцінювання результатів навчання є центральним з точки зору аналітики. У межах цього бізнес-процесу здійснюється фіксація результатів складання заліків та іспитів, збереження оцінок за окремими дисциплінами та навчальними періодами. Отримані дані використовуються для подальшого аналізу успішності студентів, визначення середніх показників та виявлення тенденцій у навчанні.

Окремим бізнес-процесом є накопичення та зберігання історичних даних. Оскільки інформація про навчальний процес змінюється з часом, система повинна забезпечувати збереження даних за кілька навчальних років, що дозволяє виконувати порівняльний аналіз динаміки показників у часовому розрізі.

Завершальним бізнес-процесом є аналітична обробка та формування звітності. На основі даних сховища та OLAP-куба користувачі системи отримують доступ до інтерактивних аналітичних звітів, які дозволяють виконувати фільтрацію, агрегування та деталізацію інформації за різними вимірами, такими як час, факультет, кафедра чи дисципліна.

Функціональні вимоги до інформаційної системи

Інформаційна система повинна забезпечувати зберігання та обробку даних, пов’язаних з діяльністю навчального закладу, з урахуванням вимог до аналізу великих обсягів інформації. Система має підтримувати повний цикл роботи з даними — від первинного обліку до аналітичної обробки та формування звітності.

Система повинна забезпечувати ведення реляційної транзакційної бази даних (OLTP), у якій зберігається інформація про студентів, факультети, кафедри, навчальні дисципліни, навчальні періоди та результати навчання. Дані в OLTP-базі повинні бути структуровані, нормалізовані та захищені за допомогою обмежень цілісності.

Обов'язковою функціональною вимогою є реалізація ETL-процесів для перенесення даних з транзакційної бази даних у сховище даних. ETL-процеси повинні включати етапи вилучення даних, їх очищення, трансформації, перевірки коректності та завантаження у багатовимірну модель. Некоректні або помилкові записи повинні оброблятися окремо з можливістю подальшого аналізу.

Система повинна забезпечувати побудову сховища даних, організованого за багатовимірною моделлю типу «зірка», що включає фактові таблиці та таблиці вимірів. Структура сховища має забезпечувати ефективне виконання аналітичних запитів і підтримку історичних даних за тривалий період часу.

Для аналітичної обробки даних система повинна підтримувати створення OLAP-куба з використанням SQL Server Analysis Services. OLAP-куб має містити набір вимірів і мір, а також підтримувати виконання багатовимірних обчислень, агрегування показників та аналіз даних у часовому розрізі.

Інформаційна система повинна забезпечувати формування аналітичних звітів із використанням SQL Server Reporting Services. Звіти мають підтримувати різні форми подання інформації, зокрема табличні, матричні та графічні звіти, а також інтерактивні панелі керування з можливістю фільтрації, сортування та деталізації даних.

Система повинна надавати користувачеві можливість виконувати аналіз даних за різними вимірами, такими як факультет, кафедра, дисципліна та навчальний період, а також забезпечувати швидкий доступ до агрегованих показників для підтримки прийняття управлінських рішень.

Бізнес-правила та обмеження предметної області

У межах предметної області інформаційної системи обліку та аналізу діяльності навчального закладу діє набір бізнес-правил, які визначають допустимі стани даних та взаємозв’язки між основними сутностями системи. Дотримання цих правил забезпечує логічну цілісність даних і коректність аналітичної обробки.

Кожен студент може бути зарахований лише до одного факультету та однієї кафедри в межах одного навчального періоду. Зміна факультету або кафедри можлива лише з початку нового навчального періоду, що дозволяє зберігати коректну історію навчання.

Кожна кафедра повинна бути пов’язана з одним факультетом, при цьому факультет може включати декілька кафедр. Дисципліна закріплюється за конкретною кафедрою та може викладатися в межах кількох навчальних періодів.

Оцінка результатів навчання виставляється студенту лише за дисципліни, які передбачені навчальним планом та закріплені за відповідним навчальним періодом. Для кожної пари «студент – дисципліна – навчальний період» може існувати лише один запис з результатом оцінювання.

Значення оцінок повинні відповідати визначеній шкалі оцінювання та не можуть виходити за встановлені межі. Записи з некоректними або відсутніми значеннями оцінок вважаються помилковими та підлягають окремій обробці в процесі ETL.

Кожен навчальний період повинен мати чітко визначені часові межі та бути однозначно ідентифікованим у системі. Дані, що відносяться до різних навчальних періодів, не можуть змішуватися в межах одного аналітичного запису.

Система повинна забезпечувати збереження історичних даних без їх фізичного видалення. Усі зміни в структурі навчального процесу або складі студентів мають відображатися шляхом додавання нових записів, що дозволяє виконувати аналіз динаміки показників у часовому розрізі.

Для забезпечення цілісності даних у реляційній базі даних застосовуються первинні та зовнішні ключі, обмеження унікальності та перевірки допустимих значень. Порушення бізнес-правил повинні виявлятися на етапі завантаження даних та враховуватися в аналітичних процесах.

Проєктування бази даних

Концептуальна модель бази даних

Концептуальна модель бази даних описує предметну область на рівні основних сутностей та зв’язків між ними без прив’язки до конкретної СУБД або фізичної реалізації. Для проєктування концептуальної моделі використано ER-підхід (Entity–Relationship), який дозволяє формалізувати структуру даних навчального закладу та визначити правила їх взаємодії.

Основними сутностями предметної області є: **Факультет**, **Кафедра**, **Дисципліна**, **Студент**, **Навчальний період** та **Результат навчання (оцінювання)**. Сутності відображають ключові об’єкти обліку, а зв’язки між ними визначають логіку навчального процесу та подальшого аналітичного використання даних.

Сутність **Факультет** описує структурний підрозділ навчального закладу та містить ідентифікатор факультету і його назву. Сутність **Кафедра** належить до певного факультету та характеризується унікальним ідентифікатором, назвою та посиланням на факультет. Таким чином, між сутностями **Факультет** і **Кафедра** встановлено зв’язок типу **1:M** (один факультет може містити багато кафедр, але кожна кафедра належить лише одному факультету).

Сутність **Дисципліна** описує навчальну дисципліну та включає ідентифікатор, назву та належність до кафедри. Зв’язок між сутностями **Кафедра** і **Дисципліна** також має тип **1:M**, оскільки одна кафедра

може викладати багато дисциплін, а кожна дисципліна закріплена за однією кафедрою.

Сутність **Студент** містить облікові та ідентифікаційні дані студента, а також належність до факультету/кафедри. У спрощеному вигляді в концептуальній моделі фіксується зв’язок між студентом і кафедрою (або факультетом) як **M:1** (багато студентів належать одній кафедрі/факультету). За потреби детальнішого обліку переходів між підрозділами цей зв’язок може бути реалізований через окрему історичну сутність, однак для базової моделі достатньо фіксації актуальної належності студента.

Сутність **Навчальний період** (семестр/навчальний рік) використовується для коректного розділення даних у часовому розрізі та містить ідентифікатор періоду, назву періоду та його часові межі. Це дозволяє накопичувати історичні дані та виконувати порівняльний аналіз успішності за різні роки або семестри.

Сутність **Результат навчання (оцінювання)** є центральною подієвою сутністю, яка фіксує факт отримання студентом оцінки з певної дисципліни у визначеному навчальному періоді. Вона встановлює зв’язки з сутностями **Студент**, **Дисципліна** та **Навчальний період**. Таким чином, зв’язки між **Студентом** і **Дисципліною** є **M:N**, але в концептуальній моделі вони реалізуються через сутність **Результат навчання**, що містить додатковий атрибут “оцінка” і забезпечує однозначність записів для комбінації «студент–дисципліна–період».

Отже, ER-діаграма відображає ієрархічну структуру навчального закладу (факультет → кафедра → дисципліни), контингент студентів та подієві дані про результати навчання, які у подальшому є основою для побудови логічної моделі та реалізації сховища даних і OLAP-аналітики.

Логічна модель бази даних

Логічна модель бази даних побудована на основі концептуальної ER-моделі та визначає структуру таблиць, їх атрибути, первинні/зовнішні ключі, а також правила зв’язків між сутностями. На цьому рівні модель уже орієнтована на реляційну СУБД та забезпечує коректність зберігання даних для подальшого завантаження у сховище даних і побудови OLAP-аналітики.

У проєкті реалізовано дві взаємопов’язані частини логічної моделі: транзакційну (OLTP) базу **HR_Variant7** для оперативного обліку кадрових подій і відпусток та аналітичну (DW) базу **HR_DW_Variant7** зі схемою типу «зірка» для швидкого аналізу. У межах цього підпункту основний акцент зроблено на OLTP-моделі та її нормалізації до 3НФ, а також коротко зафіксовано логіку DW-моделі як продовження проєктування.

Транзакційна база **HR_Variant7** містить основні таблиці: **Department**, **Employee**, **Position**, **EmployeePositionHistory**, **Vacation**, **VacationType** та допоміжну **Numbers**. Усі сутності мають сурогатні первинні ключі типу `int` або `bigint`, що забезпечує однозначну ідентифікацію записів та зручність зв’язування таблиць через зовнішні ключі.

Таблиця **Department** призначена для зберігання структури підрозділів і містить поля *DepartmentID (PK)*, *DepartmentName*, а також два зовнішні ключі: *ParentDepartmentID (FK, nullable)* для реалізації ієрархії підрозділів і *ManagerEmployeeID (FK, nullable)* для закріплення керівника підрозділу. Таким чином, **Department** має рекурсивний зв’язок «підрозділ–батьківський підрозділ» типу 1:M (один підрозділ може мати багато дочірніх), а також зв’язок із **Employee** за керівником (керівник є працівником).

Таблиця **Employee** зберігає персональні та кадрові дані працівників і включає *EmployeeID (PK)*, *EmployeeCode*, *LastName*, *FirstName*, *MiddleName*, *BirthDate*, *Gender*, *HireDate*, *TerminationDate*, *Email*, *Phone*, *Status*. Зберігання атрибутів працівника в окремій таблиці відповідає

принципам нормалізації та усуває дублювання персональних даних у подієвих таблицях (історії позицій, відпустках тощо).

Таблиця **Position** описує посади в організації та містить *PositionID (PK)*, *PositionName*, *Grade*, *BaseSalary*, *IsActive*. Винесення довідника посад в окрему таблицю забезпечує повторне використання записів (одна посада може бути призначена багатьом працівникам у різні моменти часу) та спрощує підтримку актуальності довідникових значень.

Таблиця **EmployeePositionHistory** фіксує кадрові переміщення та історію призначень, є подієвою таблицею і містить *PositionHistoryID (PK)*, а також зовнішні ключі *EmployeeID (FK)*, *DepartmentID (FK)*, *PositionID (FK)* і атрибути періоду та умов: *StartDate*, *EndDate (nullable)*, *Salary*, *ChangeReason (nullable)*. Через цю таблицю реалізовано зв’язки типу 1:M між **Employee** і **EmployeePositionHistory**, між **Department** і **EmployeePositionHistory**, а також між **Position** і **EmployeePositionHistory** (один працівник/підрозділ/посада можуть мати багато історичних записів). Логічно вважається, що активним є запис із *EndDate = NULL*, а коректність періодів забезпечується правилами бізнес-логіки (початок < кінець, відсутність некоректних перетинів за потреби).

Таблиця **VacationType** є довідником типів відпусток і включає *VacationTypeID (PK)*, *TypeName*, *IsPaid*, *DefaultDaysPerYear (nullable)*, *IsActive*. Винесення типів відпусток у довідник забезпечує нормалізоване зберігання атрибутів типу (оплачуваність, стандартна кількість днів) та підтримку єдиних правил для всіх відпусток цього типу.

Таблиця **Vacation** відображає факти надання відпусток і містить *VacationID (PK)*, зовнішні ключі *EmployeeID (FK)*, *VacationTypeID (FK)* та атрибути: *StartDate*, *EndDate*, *DaysCount*, *Status*, *CreatedAt*, *ApprovedAt (nullable)*. Зв’язки: один працівник може мати багато записів відпусток (1:M), а кожна відпустка належить одному типу відпустки (M:1). Поле

DaysCount зберігається як атрибут події (факту відпустки), що дозволяє фіксувати фактичну тривалість незалежно від календарних особливостей і забезпечує коректну аналітику.

Допоміжна таблиця **Numbers** використовується як технічний об’єкт для генерації/розгортання рядів або допоміжних операцій під час наповнення даних та ETL-процесів. Її наявність не впливає на нормалізацію предметних даних, але підвищує зручність підготовки великих наборів тестових даних.

Нормалізація OLTP-моделі виконана щонайменше до **третьої нормальної форми (3НФ)**. Довідникові сутності (**Department**, **Position**, **VacationType**) винесені в окремі таблиці, що усуває дублювання назв і параметрів у подієвих таблицях. Подієві таблиці (**EmployeePositionHistory**, **Vacation**) містять лише ключі посилань (FK) та атрибути події (дати, статус, зарплата/тривалість), тобто всі неключові атрибути функціонально залежать тільки від первинного ключа відповідної таблиці, а транзитивні залежності відсутні.

Для забезпечення продуктивності та коректності зв’язків у логічній моделі передбачено індексування ключових полів. Первинні ключі індексуються автоматично, а індекси на зовнішніх ключах доцільні для прискорення приєднань у запитах та під час ETL-завантажень, зокрема для полів *EmployeeID*, *DepartmentID*, *PositionID*, *VacationTypeID*. Додатково логічно обґрунтованими є індекси за датами (*StartDate*, *EndDate*) у таблицях **EmployeePositionHistory** та **Vacation**, оскільки часовий аналіз і вибірка періодів є типовими операціями.

Аналітична база **HR_DW_Variant7** логічно організована як схема «зірка» з вимірами **DimDate**, **DimDepartment**, **DimEmployee**, **DimPosition**, **DimVacationType** та фактами **FactEmployeeMovement** і **FactVacation**. Такий підхід відокремлює довідникові атрибути (виміри) від подієвих показників (факти) та оптимізує виконання агрегованих запитів і

побудову OLAP-куба та звітів у подальших розділах.

Отже, логічна модель забезпечує цілісне зберігання кадрових даних і подій у нормалізованій OLTP-структурі та створює основу для подальшої фізичної реалізації (DDL-скриптів, обмежень цілісності, індексів) і формування аналітичного контуру (DW/SSAS/SSRS).

Фізична реалізація

Фізична реалізація бази даних виконана у СУБД Microsoft SQL Server та включає створення таблиць, визначення типів даних, первинних і зовнішніх ключів, обмежень цілісності (constraints), а також індексів для підвищення продуктивності запитів. На фізичному рівні структура OLTP-бази **HR_Variant7** реалізує нормалізовану модель (3НФ) для оперативного обліку, а аналітична база **HR_DW_Variant7** містить виміри та факти, необхідні для OLAP-куба і звітності.

Для забезпечення сутнісної цілісності у всіх основних таблицях визначено первинні ключі (PRIMARY KEY). Для забезпечення посилальної цілісності між сутностями використано зовнішні ключі (FOREIGN KEY), зокрема для зв’язків між працівниками, підрозділами, посадами, історією призначень та відпустками. Додатково застосовано обмеження NOT NULL для критично важливих атрибутів та рекомендовані перевірки (CHECK) для контролю допустимих значень, наприклад для тривалості відпустки та коректності дат.

Для оптимізації запитів та прискорення операцій приєднання таблиць створено індекси на зовнішніх ключах і полях дат, які часто використовуються у фільтрації та часовому аналізі. Первинні ключі індексуються автоматично, а додаткові некластерні індекси (NONCLUSTERED) додаються на поля *EmployeeID*, *DepartmentID*, *PositionID*, *VacationTypeID*, а також на *StartDate/EndDate* у подієвих таблицях.

Тригери та збережені процедури у базовій реалізації можуть не

використовуватися, оскільки контроль якості та перевірка “good rows / bad rows” реалізовані на рівні ETL-процесів (SSIS). За потреби допустимо додати процедури для сервісних операцій або тригери для аудиту змін, однак у межах курсового проєкту достатньо реалізації через constraints та ETL-логіку.

Генерація та наповнення даних

Для забезпечення коректної перевірки роботи транзакційної та аналітичної частин системи було реалізовано процес генерації та наповнення баз даних значним обсягом тестових даних. Генерація даних виконувалась з урахуванням структури предметної області, бізнес-правил та обмежень цілісності, що дозволило отримати наближені до реальних дані для подальшого аналізу.

Основним інструментом наповнення даних виступає Microsoft SQL Server, з використанням SQL-скриптів, вбудованих можливостей СУБД та допоміжних таблиць. Для масового створення записів застосовувались цикли, функції генерації псевдовипадкових значень, а також допоміжна таблиця **Numbers**, яка дозволяє швидко формувати великі набори даних без залучення зовнішніх генераторів.

Генерація довідникових даних (факультети/підрозділи, посади, типи відпусток) виконувалась у першу чергу. Такі дані мають відносно статичний характер і формують основу для подальшого наповнення подієвих таблиць. Значення атрибутів довідників створювались з урахуванням логіки предметної області, наприклад для посад задавались базові оклади та рівні (grade), а для типів відпусток — ознака оплачуваності та стандартна кількість днів.

Дані про працівників генерувались із використанням унікальних ідентифікаторів та кодів співробітників, а також випадково сформованих персональних атрибутів (дата народження, дата прийому на роботу, статус). При цьому дотримувались логічні обмеження, зокрема дата звільнення не може передувати даті прийому, а статус працівника відповідає наявності або

відсутності дати звільнення.

Наповнення таблиці **EmployeePositionHistory** здійснювалось із моделюванням реальних кадрових подій. Для кожного працівника формувався один або кілька записів історії призначень із коректними часовими інтервалами, змінами підрозділу, посади та рівня заробітної плати. Такий підхід дозволив створити достатньо великий і різноманітний набір подій для подальшого аналізу кадрових переміщень у сховищі даних.

Дані про відпустки у таблиці **Vacation** генерувались з урахуванням типів відпусток, тривалості, статусу та періодів часу. Для кожного працівника могло бути створено декілька записів відпусток у різні періоди, що дозволяє аналізувати як загальну кількість відпусток, так і їх тривалість та розподіл за типами.

Після наповнення транзакційної бази даних виконувалось завантаження інформації у сховище даних **HR_DW_Variant7** за допомогою ETL-процесів, реалізованих у SQL Server Integration Services. У межах ETL забезпечувалась перевірка коректності значень, фільтрація помилкових записів та контроль кількості оброблених рядків, що дозволяє гарантувати якість даних, використаних для аналітики.

Загальний підхід до генерації та наповнення даних дозволив сформувати обсяг інформації, достатній для демонстрації роботи ETL-процесів, OLAP-куба та аналітичних звітів, а також для аналізу продуктивності системи при обробці великих наборів даних.

Обсяги згенерованих даних та перевірка якості

Після реалізації процесу генерації даних транзакційна база **HR_Variant7** була наповнена значним обсягом тестових записів, що дозволяє імітувати реальні умови експлуатації інформаційної системи та виконувати аналітичні запити на великих наборах даних. Обсяги згенерованої інформації

підібрані таким чином, щоб забезпечити достатнє навантаження на ETL-процеси, сховище даних і OLAP-куб.

Довідникові таблиці містять відносно невелику, але репрезентативну кількість записів. Зокрема, таблиця **Department** включає декілька десятків підрозділів із багаторівневою ієрархією, таблиця **Position** — набір посад різних рівнів, а таблиця **VacationType** — основні типи відпусток з різними параметрами оплачуваності та тривалості. Такі обсяги є типовими для довідникових даних і практично не впливають на продуктивність, проте визначають структуру всієї системи.

Таблиця **Employee** наповнена великою кількістю записів, що моделюють кадровий склад організації за кілька років. Кількість працівників становить десятки або сотні тисяч записів, що дозволяє перевірити коректність роботи зв’язків, індексів та продуктивність запитів при об’єднанні таблиць.

Подієві таблиці мають найбільший обсяг даних. Таблиця **EmployeePositionHistory** містить кілька сотень тисяч записів, оскільки для кожного працівника зберігається історія кадрових переміщень, змін посад, підрозділів та рівнів заробітної плати. Аналогічно таблиця **Vacation** містить значну кількість записів, що відображають використання відпусток працівниками у різні часові періоди.

Під час завантаження даних у сховище **HR_DW_Variant7** формується аналітичний набір даних, де подієві записи з OLTP-бази агрегуються у фактові таблиці **FactEmployeeMovement** та **FactVacation**, а довідникові дані перетворюються на виміри **DimEmployee**, **DimDepartment**, **DimPosition**, **DimVacationType** та **DimDate**. Обсяги фактових таблиць у сховищі даних сягають сотень тисяч і більше записів, що відповідає вимогам до надвеликих баз даних у межах курсової роботи.

Перевірка якості згенерованих даних здійснювалась на декількох рівнях. На рівні транзакційної бази застосовувались обмеження цілісності (PRIMARY KEY,

FOREIGN KEY, CHECK, NOT NULL), які не дозволяють зберігати логічно некоректні або неповні записи. Це гарантує відповідність даних базовим бізнес-правилам предметної області.

На етапі ETL-процесів реалізовано додатковий контроль якості даних. У процесі завантаження виконувалась перевірка коректності дат, наявності обов'язкових зовнішніх ключів та допустимості числових значень. Записи, що не відповідали встановленим правилам, спрямовувались у окремі потоки помилок та реєструвались у службових таблицях журналювання ETL, що дозволяє аналізувати причини відхилень.

Додатково здійснювалась перевірка повноти даних шляхом порівняння кількості рядків у вихідних OLTP-таблицях та відповідних фактових і вимірних таблицях у сховищі даних. Такий підхід дозволяє впевнитися, що всі коректні записи були успішно перенесені та доступні для аналітичної обробки.

Отже, згенеровані обсяги даних та реалізовані механізми перевірки якості забезпечують достовірність аналітичної інформації, коректну роботу OLAP-куба та формування звітів, а також демонструють готовність системи до роботи з великими наборами даних у реальних умовах експлуатації.

Реалізація ETL-процесів

Архітектура сховища даних

Для забезпечення ефективної аналітичної обробки кадрових даних у проєкті реалізовано сховище даних **HR_DW_Variant7**, побудоване за багатовимірною архітектурою. Основною метою проєктування сховища даних є відокремлення транзакційного навантаження від аналітичного та оптимізація виконання агрегованих запитів, що використовуються в OLAP-кубі та звітах.

Архітектура сховища даних реалізована за схемою типу **«зірка» (star schema)**. У межах цієї схеми центральне місце займають фактові таблиці, які зберігають подієві та кількісні показники, а навколо них розташовані таблиці

вимірів, що містять описові атрибути для аналізу даних у різних розрізах. Такий підхід є оптимальним для OLAP-аналізу, оскільки забезпечує просту структуру зв’язків та високу продуктивність запитів.

У сховищі даних реалізовано **дві фактові таблиці**, що відповідають ключовим бізнес-подіям предметної області:

- FactEmployeeMovement — факти кадрових переміщень та призначень працівників;
- FactVacation — факти використання відпусток працівниками.

Фактова таблиця FactEmployeeMovement формується на основі транзакційної таблиці EmployeePositionHistory та містить показники, пов’язані з кадровими змінами. До таких показників належать кількість переміщень, рівень заробітної плати на момент призначення, а також часові характеристики події. Ця таблиця використовується для аналізу динаміки кадрових змін, структури персоналу та руху працівників між підрозділами й посадами.

Фактова таблиця FactVacation формується на основі таблиці Vacation і зберігає кількісні показники, пов’язані з відпустками працівників, зокрема кількість днів відпустки та кількість фактів надання відпусток. Вона дозволяє аналізувати використання відпусток у часовому розрізі, за типами відпусток, підрозділами та окремими працівниками.

Для забезпечення багатовимірного аналізу у сховищі даних створено п’ять вимірів, які використовуються спільно обома фактовими таблицями:

- DimDate — вимір часу, що дозволяє виконувати аналіз за датами, місяцями, кварталами та роками;
- DimDepartment — вимір підрозділів, який відображає організаційну структуру та підтримує ієрархію підрозділів;
- DimEmployee — вимір працівників з основними атрибутами персоналу;
- DimPosition — вимір посад, що використовується для аналізу кадрової

структури;

- DimVacationType — вимір типів відпусток, який застосовується для класифікації та порівняння відпусток за ознаками оплачуваності та призначення.

Кожна фактова таблиця містить зовнішні ключі, що посилаються на відповідні виміри, а також числові показники, які агрегуються під час аналізу. Зв’язки між фактами та вимірами мають тип M:1, що відповідає класичній схемі «зірка» та спрощує побудову запитів у OLAP-кубі.

ETL-процеси для наповнення сховища даних реалізовані за допомогою SQL Server Integration Services (SSIS) та організовані за принципом поетапного завантаження. На першому етапі виконується завантаження таблиць вимірів, що дозволяє сформувати повний набір ключів та атрибутів, необхідних для подальшого завантаження фактових даних. На другому етапі здійснюється завантаження фактових таблиць з використанням відповідних сурогатних ключів вимірів.

Для забезпечення цілісності та узгодженості даних у процесі ETL використовується послідовність завантаження «виміри → факти», а також контроль відповідності ключів між транзакційною базою та сховищем даних. Така архітектура дозволяє масштабувати систему, спрощує розширення набору вимірів або показників і створює надійну основу для подальшої реалізації OLAP-куба та аналітичних звітів.

Опис пакетів

ETL-процеси у проєкті реалізовані за класичною трирівневою схемою **Extract → Transform → Load**, що забезпечує поетапну обробку даних, контроль їх якості та коректне завантаження у сховище даних **HR_DW_Variant7**. Для реалізації ETL використано інструмент **SQL Server Integration Services (SSIS)**, який дозволяє організувати потоки даних,

трансформації та обробку помилок у наочному та керованому вигляді.

На етапі **Extract (вилучення даних)** виконується отримання інформації з транзакційної бази даних **HR_Variant7**. Джерелами даних є основні OLTP-таблиці, зокрема **Employee**, **Department**, **Position**, **EmployeePositionHistory**, **Vacation** та **VacationType**. Вилучення даних здійснюється за допомогою компонентів *OLE DB Source* з використанням SQL-запитів, які дозволяють обмежувати обсяг вибірки та відбирати лише актуальні або змінені записи. На цьому етапі не виконуються складні перетворення, що дозволяє зберегти первинний вигляд даних для подальшої обробки.

Етап **Transform (трансформація даних)** є центральним у процесі ETL та включає очищення, перевірку та підготовку даних до аналітичного використання. У потоках даних застосовуються різні типи трансформацій SSIS. За допомогою компонента *Derived Column* виконуються обчислення нових атрибутів, приведення типів даних та формування технічних полів, необхідних для сховища даних. Компонент *Lookup* використовується для зіставлення записів з таблицями вимірів та отримання відповідних сурогатних ключів. Для виявлення некоректних або неповних записів застосовується *Conditional Split*, який розділяє потік на коректні дані (Good Rows) та помилкові записи (Error Rows).

У процесі трансформації також здійснюється перевірка бізнес-правил, зокрема коректності дат (дата початку не перевищує дату завершення), наявності обов’язкових зовнішніх ключів та допустимості числових значень. Для контролю обсягу оброблених даних використовується компонент *Row Count*, який дозволяє відстежувати кількість рядків, що пройшли через кожен етап ETL, та порівнювати їх з очікуваними значеннями.

Записи, що не відповідають встановленим вимогам, спрямовуються у окремі потоки помилок. Такі дані не завантажуються у сховище, а зберігаються у

спеціальних таблицях або файлах журналювання, що дозволяє аналізувати причини помилок та вдосконалювати правила очищення даних.

На етапі **Load (завантаження даних)** підготовлені та перевірені дані завантажуються у таблиці вимірів та фактові таблиці сховища даних **HR_DW_Variant7**. Завантаження вимірів виконується перед завантаженням фактів, що забезпечує наявність усіх необхідних сурогатних ключів. Для завантаження використовується компонент *OLE DB Destination* у режимі пакетної вставки, що підвищує продуктивність при роботі з великими обсягами даних.

Завантаження фактових таблиць здійснюється з використанням зовнішніх ключів на виміри та числових показників, що формують основу для подальшої агрегації. Після завершення завантаження виконується фіксація результатів ETL-процесу у службових таблицях журналювання, що містять інформацію про дату запуску, кількість оброблених рядків та статус виконання пакета.

Таким чином, реалізована схема Extract → Transform → Load забезпечує надійне та кероване перенесення даних з OLTP-бази у сховище даних, гарантує якість аналітичної інформації та створює основу для побудови OLAP-куба і аналітичних звітів у наступних етапах проєкту.

Трансформації у ETL-процесах

У процесі реалізації ETL-процесів у SQL Server Integration Services було використано кілька типів стандартних трансформацій, що забезпечують очищення, перевірку та підготовку даних до завантаження у сховище даних. Застосування різних трансформацій дозволяє реалізувати бізнес-правила предметної області, підвищити якість даних та забезпечити коректну аналітичну обробку.

Трансформація **Derived Column** використовується для створення нових атрибутів і модифікації існуючих полів без зміни вихідних даних. За допомогою

цієї трансформації виконуються обчислення технічних полів, приведення дат до необхідного формату, формування службових ознак, а також обчислення похідних значень, наприклад тривалості відпустки або статусу запису на основі вхідних даних.

Трансформація **Lookup** застосовується для зіставлення даних з транзакційної бази з відповідними таблицями вимірів у сховищі даних. Вона дозволяє отримувати сурогатні ключі вимірів на основі бізнес-ключів, таких як ідентифікатор працівника, підрозділу, посади або типу відпустки. У випадку відсутності відповідного запису у вимірі трансформація дозволяє ідентифікувати помилкові або нові дані та передати їх у відповідний потік обробки.

Трансформація **Conditional Split** використовується для розділення потоків даних залежно від виконання заданих умов. З її допомогою реалізовано логіку відбору коректних записів (Good Rows) та виявлення помилкових або неповних даних (Error Rows). Наприклад, записи з некоректними датами або відсутніми обов'язковими ключами спрямовуються у окремий потік для подальшого аналізу.

Трансформація **Data Conversion** застосовується для приведення типів даних до форматів, сумісних зі схемою сховища даних. Це особливо важливо під час завантаження даних з OLTP-бази, де типи можуть відрізнятися від типів у вимірах або фактових таблицях. Перетворення типів дозволяє уникнути помилок завантаження та забезпечує узгодженість структури даних.

Трансформація **Aggregate** використовується для попереднього агрегування даних перед завантаженням у фактові таблиці. За її допомогою виконуються обчислення підсумкових показників, таких як загальна кількість кадрових переміщень або сумарна кількість днів відпусток за певний період. Попереднє агрегування зменшує обсяг завантажуваних даних та підвищує продуктивність аналітичних запитів.

Окрім зазначених трансформацій, у ETL-процесах також застосовуються

допоміжні компоненти, зокрема **Row Count** для підрахунку кількості оброблених рядків та контролю повноти завантаження, а також механізми обробки помилок, що дозволяють журналювати відхилення та підвищувати надійність процесу.

Отже, використання щонайменше п’яти типів трансформацій у SSIS забезпечує повноцінну реалізацію ETL-процесів, контроль якості даних та підготовку коректної інформаційної бази для побудови OLAP-куба і аналітичних звітів.

Обробка помилок, логування та потоки

У процесі реалізації ETL-процесів особливу увагу приділено обробці помилок та контролю якості даних, оскільки коректність аналітичної інформації безпосередньо залежить від надійності етапу завантаження. Для цього в SSIS реалізовано механізм розділення потоків даних на коректні та помилкові записи з подальшим журналюванням результатів виконання пакетів.

Обробка помилок здійснюється на етапі трансформації даних із використанням стандартних компонентів SSIS. За допомогою трансформації **Conditional Split** вхідний потік розділяється на два основні напрями: **Good Rows** та **Bad Rows**. До потоку Good Rows потрапляють записи, що відповідають встановленим бізнес-правилам та обмеженням цілісності, зокрема мають коректні дати, допустимі числові значення та всі необхідні зовнішні ключі.

Записи, які не проходять перевірки, спрямовуються до потоку Bad Rows. До таких записів належать дані з некоректними або відсутніми значеннями ключових полів, порушенням логіки дат (наприклад, дата завершення раніше дати початку), а також записи, для яких не знайдено відповідних значень у таблицях вимірів під час виконання операцій Lookup. Відокремлення помилкових даних дозволяє уникнути зупинки всього ETL-процесу через окремі некоректні записи.

Потік Good Rows використовується для подальшого завантаження даних у

таблиці вимірів та фактові таблиці сховища даних. Перед завантаженням може виконуватися додаткове перетворення типів та агрегування, після чого дані передаються до компонентів OLE DB Destination для запису у **HR_DW_Variant7**.

Потік Bad Rows спрямовується у спеціальні таблиці або файли журналювання, призначені для зберігання інформації про помилкові записи. Разом з основними атрибутами даних до таких таблиць додається службова інформація, зокрема опис причини помилки, дата та час обробки, а також ідентифікатор ETL-пакета. Це дозволяє виконувати подальший аналіз помилок та вдосконалювати правила очищення даних.

Для контролю виконання ETL-процесів у пакети включено механізми логування. За допомогою компонентів **Row Count** фіксується кількість рядків, що проходять через кожен ключовий етап обробки. Отримані значення використовуються для порівняння кількості вхідних даних з кількістю завантажених записів у сховищі, що дозволяє виявляти втрати або аномалії у процесі завантаження.

Додатково у процесі логування зберігається інформація про статус виконання ETL-пакетів, час початку та завершення обробки, а також можливі помилки під час виконання. Журнали виконання дозволяють відстежувати історію запусків ETL, оцінювати стабільність процесів та швидко локалізувати проблемні ділянки у разі збоїв.

Таким чином, реалізована система обробки помилок, логування та розділення потоків “Good Rows / Bad Rows” забезпечує надійність ETL-процесів, підвищує якість даних у сховищі та створює прозорий механізм контролю і аналізу результатів завантаження.

OLAP-куб та аналітичні звіти

DSV, виміри та міри

Для забезпечення багатовимірного аналізу даних у проєкті створено OLAP-куб із використанням **SQL Server Analysis Services (SSAS)** на основі сховища даних **HR_DW_Variant7**. OLAP-куб дозволяє виконувати швидкий аналіз великих обсягів інформації, агрегувати показники та досліджувати дані у різних аналітичних розрізах.

Початковим етапом побудови куба є створення **Data Source View (DSV)**, який відображає логічну модель сховища даних у середовищі SSAS. У DSV включено таблиці вимірів та фактові таблиці сховища даних, а також визначено зв’язки між ними відповідно до схеми «зірка». Використання DSV дозволяє ізолювати структуру OLAP-куба від фізичної реалізації бази даних та спрощує подальше налаштування вимірів і мір.

У межах OLAP-куба реалізовано **п’ять основних вимірів**, які забезпечують аналіз даних за ключовими аспектами предметної області:

- DimDate — вимір часу, який містить ієрархію Рік → Квартал → Місяць → День та використовується для часової аналітики;
- DimDepartment — вимір підрозділів, що відображає організаційну структуру та дозволяє аналізувати дані за відділами та їх ієрархією;
- DimEmployee — вимір працівників із базовими персональними атрибутами, що використовується для детального аналізу кадрових показників;
- DimPosition — вимір посад, який дозволяє аналізувати структуру персоналу за посадами та рівнями;
- DimVacationType — вимір типів відпусток, який забезпечує класифікацію відпусток за призначенням та ознакою оплачуваності.

Кожен вимір має чітко визначений ключовий атрибут та набір описових

атрибутів, що використовуються для фільтрації, групування та побудови ієрархій у звітах. Виміри пов’язані з фактовими таблицями через відповідні сурогатні ключі, що забезпечує коректну роботу багатовимірної моделі.

На основі фактових таблиць FactEmployeeMovement та FactVacation у OLAP-кубі сформовано набір мір, які відображають кількісні показники предметної області. У кубі реалізовано щонайменше сім мір, зокрема:

- Employee Movements Count — кількість кадрових переміщень;
- Active Employees Count — кількість активних працівників;
- Average Salary — середній рівень заробітної плати;
- Total Salary — сумарний фонд оплати праці;
- Vacations Count — кількість відпусток;
- Total Vacation Days — загальна кількість днів відпусток;
- Average Vacation Duration — середня тривалість відпустки.

Міри налаштовані з відповідними типами агрегування (SUM, COUNT, AVG), що дозволяє виконувати коректні аналітичні обчислення на різних рівнях деталізації. Для деяких показників використовується обчислення на основі базових мір, що підвищує гнучкість аналітики та зменшує дублювання логіки.

OLAP-куб підтримує багатовимірний аналіз даних за всіма вимірами одночасно, що дозволяє, наприклад, аналізувати динаміку кадрових переміщень у часовому розрізі, порівнювати використання відпусток між підрозділами або оцінювати середній рівень заробітної плати за посадами та роками.

Таким чином, створений OLAP-куб на основі DSV, п’яти вимірів та набору мір формує повноцінну аналітичну модель, яка слугує основою для побудови аналітичних звітів у SQL Server Reporting Services та підтримує прийняття управлінських рішень на основі агрегованих і деталізованих даних.

MDX-обчислення та елементи Time Intelligence, Ranking і KPI

Для розширення аналітичних можливостей OLAP-куба у проєкті використано обчислювані міри та вирази **MDX (Multidimensional Expressions)**. MDX-обчислення дозволяють формувати похідні показники, виконувати часовий аналіз, ранжування та оцінювання ефективності без зміни структури сховища даних. Це підвищує гнучкість аналітики та дозволяє реалізувати складні бізнес-метрики на рівні куба.

Одним з базових MDX-обчислень є **середній рівень заробітної плати**, який розраховується на основі сумарного фонду оплати праці та кількості активних працівників. Дана міра дозволяє аналізувати середню зарплату у розрізі підрозділів, посад і часових періодів без необхідності попередньої агрегації у фактовій таблиці.

Наступним MDX-обчисленням є **кількість активних працівників**, яка визначається шляхом підрахунку унікальних співробітників із актуальним статусом. Це обчислення використовується як самостійна аналітична міра, а також як базова складова для інших показників, зокрема середньої зарплати та коефіцієнтів навантаження.

Третє MDX-обчислення реалізує **середню тривалість відпустки**, яка визначається як відношення загальної кількості днів відпусток до кількості фактів надання відпусток. Дана міра дозволяє порівнювати використання відпусток між різними підрозділами, типами відпусток та часовими періодами.

Четверте MDX-обчислення пов’язане з аналізом кадрових змін і визначає **частоту кадрових переміщень**, тобто середню кількість переміщень на одного працівника за вибраний період. Такий показник використовується для оцінки стабільності персоналу та кадрової динаміки в організації.

П’яте MDX-обчислення реалізує **відсоткову частку відпусток певного типу** у загальній кількості відпусток. Воно дозволяє аналізувати структуру використання відпусток і визначати, які типи відпусток є найбільш поширеними

у різні періоди часу.

Для реалізації **Time Intelligence** у кубі використовується вимір **DimDate** з ієрархією «Рік → Квартал → Місяць → День». На основі цього виміру створено MDX-обчислення для аналізу динаміки показників у часі, зокрема:

- порівняння поточного значення показника з аналогічним періодом попереднього року;
- обчислення приросту або зниження показників у відсотках;
- накопичувальні підсумки (YTD — Year-To-Date) для заробітної плати та днів відпусток.

Елементи Ranking реалізовані за допомогою MDX-функцій ранжування, що дозволяють впорядковувати підрозділи або посади за певними показниками, наприклад за сумарним фондом оплати праці, кількістю працівників або загальною тривалістю відпусток. Ранжування використовується у звітах для виділення підрозділів з найбільшими або найменшими значеннями показників.

Для візуального оцінювання ефективності в OLAP-кубі реалізовано KPI (Key Performance Indicators). KPI визначають цільові значення для ключових показників, таких як середня заробітна плата, кількість днів відпусток на одного працівника або рівень кадрових переміщень. Для кожного KPI задано базову міру, цільове значення та статус (норма, відхилення, критичне відхилення), що дозволяє швидко оцінювати стан показників у звітах.

Отже, використання MDX-обчислень у поєднанні з Time Intelligence, Ranking та KPI значно розширює аналітичні можливості OLAP-куба, забезпечує глибокий аналіз даних у часовому та структурному розрізах і створює потужну основу для побудови інформативних аналітичних звітів у SQL Server Reporting Services.

Аналітичні звіти

Для візуалізації результатів багатовимірного аналізу та надання користувачам зручного доступу до аналітичної інформації у проєкті реалізовано комплекс звітів із використанням **SQL Server Reporting Services (SSRS)**. Звіти побудовані на основі OLAP-куба та MDX-запитів і охоплюють різні типи подання даних, що відповідає вимогам до повноцінної аналітичної системи.

У межах проєкту реалізовано **п’ять основних типів звітів**, які забезпечують як детальний, так і узагальнений аналіз кадрових показників.

Табличний звіт (Table Report) використовується для подання детальної інформації у вигляді списків і рядків. У даному проєкті табличний звіт відображає показники по працівниках або підрозділах, зокрема кількість кадрових переміщень, сумарну заробітну плату та кількість днів відпусток. Табличний формат дозволяє зручно переглядати точні значення показників, застосовувати сортування та фільтрацію за вимірами.

Матричний звіт (Matrix Report) застосовується для перехресного аналізу даних у двовимірному представленні. У матричному звіті показники відображаються у розрізі рядків і стовпців, наприклад підрозділи по рядках та роки або місяці по стовпцях. Такий тип звіту дозволяє швидко порівнювати значення показників між різними категоріями та виявляти тенденції у часовому розрізі.

Графічні звіти (Chart Reports) використовуються для наочного відображення динаміки та структури даних. У проєкті реалізовано щонайменше три типи діаграм:

- стовпчикові діаграми для порівняння показників між підрозділами або посадами;
- лінійні діаграми для аналізу змін показників у часі;
- кругові діаграми для відображення структури, наприклад розподілу відпусток за типами.

Графічні звіти дозволяють швидко оцінювати тенденції та співвідношення між показниками без необхідності аналізу великих таблиць чисел.

Інтерактивна панель керування (Dashboard) об’єднує ключові показники у єдиному звіті та забезпечує швидкий огляд стану кадрових процесів. Dashboard містить зведені міри, графіки та елементи візуального контролю (KPI), що відображають поточні значення показників, їх відхилення від цільових значень та загальну динаміку. Такий звіт використовується для оперативного аналізу та підтримки управлінських рішень.

Звіт з деталізацією (Drill-down Report) реалізує можливість переходу від узагальнених показників до детальних даних. Користувач може розгортати ієрархії вимірів або переходити від зведених значень до детальних записів, наприклад від показників по факультету до окремих підрозділів або працівників. Drill-down функціональність забезпечує глибокий аналіз даних без створення окремих звітів для кожного рівня деталізації.

Усі звіти підтримують параметризацію, зокрема вибір часових періодів, підрозділів, посад або типів відпусток. Використання параметрів дозволяє адаптувати звіти під потреби користувача та виконувати аналіз даних у довільних розрізах.

Таким чином, реалізований набір звітів у SSRS охоплює всі основні типи аналітичного подання інформації та забезпечує інтерактивний, наочний і гнучкий доступ до даних OLAP-куба, що повністю відповідає вимогам курсового проєкту.

Параметри звітів

Для забезпечення інтерактивності та гнучкого аналізу даних у звітах SSRS у проєкті реалізовано систему параметрів, яка дозволяє користувачеві змінювати контекст відображення показників без редагування звіту. Параметри використовуються у MDX-запитах до OLAP-куба та впливають на фільтрацію, групування і рівень деталізації результатів.

Параметри типу **dropdown (одиночний вибір)** застосовуються у випадках, коли користувач повинен обрати одне значення з довідника. Наприклад, у звітах може використовуватися вибір конкретного року, місяця, посади або типу відпустки. Dropdown-параметри роблять навігацію простою та зменшують ризик помилкового введення значень, оскільки користувач обирає лише з дозволеного переліку.

Параметри типу **multiselect (множинний вибір)** використовуються для сценаріїв, коли необхідно аналізувати дані одночасно за кількома значеннями виміру. Наприклад, користувач може обрати декілька підрозділів для порівняння їх показників або декілька типів відпусток для аналізу структури. Multiselect-параметри підвищують аналітичну гнучкість звітів і дозволяють формувати порівняльні вибірки в межах одного запуску звіту.

Кожен параметр у звіті має визначені значення за замовчуванням, що дозволяє запускати звіти без додаткового налаштування. Наприклад, за замовчуванням може обиратися поточний рік, активні підрозділи або всі типи відпусток. Це забезпечує швидке отримання результатів та робить звіти зручними для повторного використання.

Таким чином, реалізація параметрів dropdown, multiselect, date range та cascading у звітах SSRS забезпечує інтерактивність, гнучку фільтрацію і можливість побудови різних аналітичних зрізів даних, що суттєво підвищує практичну цінність створеної системи звітності.

Аналіз продуктивності

Порівняння OLTP та OLAP

Для оцінювання ефективності реалізованої аналітичної системи було виконано аналіз продуктивності при роботі з транзакційною базою даних (OLTP) та сховищем даних з OLAP-кубом. Метою аналізу є демонстрація переваг використання сховища даних і багатовимірного аналізу для виконання складних

аналітичних запитів на великих обсягах інформації.

У якості першого тестового сценарію було обрано запит на визначення **загальної кількості днів відпусток за кожним підрозділом за обраний період часу**. В OLTP-базі такий запит потребує об’єднання кількох таблиць (**Vacation**, **Employee**, **Department**, **VacationType**), застосування фільтрів за датами та агрегації значень. При роботі з великим обсягом даних виконання такого запиту займає відчутний час через складні JOIN-операції та обробку значної кількості рядків.

Аналогічний запит в OLAP-середовищі виконується над попередньо агрегованими даними у фактовій таблиці **FactVacation** з використанням відповідних вимірів. Завдяки наявності агрегованих мір та оптимізованій структурі «зірка» час виконання запиту значно зменшується, а результат повертається практично миттєво навіть при аналізі даних за декілька років.

Другим тестовим сценарієм є запит на **аналіз середньої заробітної плати за посадами у часовому розрізі**. В OLTP-базі для цього необхідно обробляти таблицю **EmployeePositionHistory**, виконувати групування за посадами та періодами, а також розрахунок середніх значень. Час виконання такого запиту зростає пропорційно обсягу історичних даних.

У випадку OLAP-куба відповідний показник реалізований у вигляді MDX-міри, що базується на попередньо підготовлених фактах і вимірі часу. Завдяки використанню агрегатів і багатовимірної структури запит виконується значно швидше та дозволяє миттєво змінювати розрізи аналізу без повторного обчислення базових даних.

Результати порівняння показують, що виконання складних аналітичних запитів у OLAP-середовищі є у кілька разів швидшим, ніж у транзакційній базі даних. Це підтверджує доцільність відокремлення аналітичного навантаження від OLTP-бази та використання сховища даних для підтримки управлінських рішень.

Вплив індексів та агрегацій

Для підвищення продуктивності роботи з OLTP-базою було використано індексування первинних і зовнішніх ключів, а також індекси на полях дат, які часто застосовуються у фільтрації та об’єднанні таблиць. Наявність індексів дозволяє скоротити час виконання SELECT-запитів, проте при зростанні обсягів даних ефективність транзакційної бази для аналітичних задач залишається обмеженою через необхідність обробки великої кількості рядків та JOIN-операцій.

У сховищі даних основний вплив на продуктивність забезпечується не лише індексами, а й використанням **попередніх агрегацій** та структури OLAP-куба. Агрегування показників на різних рівнях вимірів (рік, місяць, підрозділ, посада) дозволяє значно скоротити обсяг обчислень під час виконання запитів. OLAP-движок використовує ці агрегати автоматично, що забезпечує стабільну та високу швидкість обробки запитів незалежно від складності аналітичного розрізу.

Таким чином, результати аналізу продуктивності демонструють, що індексування є важливим інструментом оптимізації OLTP-бази, проте для складної аналітики ключову роль відіграють сховище даних, попередні агрегації та OLAP-куб. Використання OLAP-підходу забезпечує значний виграш у швидкодії, масштабованість та зручність виконання багатовимірного аналізу на великих обсягах даних.

Тестування та валідація

Тестування та валідація реалізованої інформаційно-аналітичної системи виконувались з метою підтвердження коректності структури баз даних, правильності роботи ETL-процесів, узгодженості даних у сховищі та правильного функціонування аналітичних звітів. Перевірки проводились на всіх етапах побудови рішення: OLTP-база → ETL (SSIS) → DW → OLAP-куб → SSRS.

Перевірка обмежень цілісності

Перевірка обмежень цілісності виконувалась на рівні транзакційної бази даних **HR_Variant7** та включала контроль коректності первинних ключів, зовнішніх ключів, обмежень NOT NULL, UNIQUE та CHECK. Верифікація первинних ключів полягала у підтвердженні унікальності записів у кожній таблиці та неможливості вставки дубльованих значень ключів.

Для зовнішніх ключів перевірялась відсутність “висячих” посилань, тобто неможливість створити записи у подієвих таблицях (**Vacation**, **EmployeePositionHistory**) без відповідних записів у довідникових таблицях (**Employee**, **Department**, **Position**, **VacationType**). Такий контроль гарантує узгодженість даних і правильність подальшого завантаження у сховище даних.

Додатково перевірялись обмеження CHECK, які контролюють допустимі значення та логічні правила, наприклад коректність дат (дата завершення не може бути меншою за дату початку), додатність тривалості відпустки, а також допустимі значення ознак (наприклад IsActive, IsPaid). Для перевірки constraints виконувались тестові вставки записів з навмисними помилками та підтверджувалось, що СУБД блокує такі операції.

Перевірка коректності ETL-трансформацій

Перевірка ETL-процесів здійснювалась у середовищі **SSIS** з контролем коректності основних етапів Extract → Transform → Load. На етапі вилучення даних перевірялась відповідність вибірок даних очікуваному обсягу та правильність приєднань (у випадку використання джерел із JOIN).

На етапі трансформації особлива увага приділялась перевірці логіки трансформацій **Derived Column**, **Lookup**, **Conditional Split**, **Data Conversion** та **Aggregate**. Для Derived Column перевірялась правильність обчислення похідних атрибутів та службових полів. Для Data Conversion

перевірялась відсутність помилок приведення типів та відповідність форматів полів схемі сховища даних.

Для Lookup перевірялась коректність зіставлення бізнес-ключів із сурогатними ключами вимірів та правильна обробка ситуацій “No Match”. Записи, що не знаходили відповідності у вимірах, повинні або спрямовуватись у потік помилок, або оброблятись згідно з прийнятою логікою (наприклад через завантаження “Unknown” елементів у виміри).

Для Conditional Split тестувались умови відбору “Good Rows / Bad Rows”. Перевірялося, що записи з некоректними датами, відсутніми обов’язковими значеннями або іншими порушеннями правил спрямовуються у потік Bad Rows та не потрапляють у таблиці сховища. Для підтвердження повноти завантаження використовувались лічильники **Row Count** та порівняння кількості рядків на вході та виході кожного критичного етапу.

Після завершення завантаження виконувалась валідація даних у сховищі: перевірялась наявність ключових записів у вимірах, коректність зв’язків у фактових таблицях та відповідність агрегованих показників очікуваним значенням (контрольні суми, підрахунок кількості фактів, сумарні дні відпусток тощо).

Висновки

У межах даної курсової роботи було розроблено та реалізовано повноцінну інформаційно-аналітичну систему на основі надвеликих баз даних з використанням платформи Microsoft SQL Server. Робота охопила всі основні етапи побудови аналітичного рішення — від аналізу предметної області та проєктування реляційної бази даних до створення сховища даних, OLAP-куба та комплексу інтерактивних аналітичних звітів.

На початковому етапі виконано аналіз предметної області та визначено ключові бізнес-процеси кадрового обліку, зокрема управління структурою

підрозділів, облік працівників, кадрові переміщення та використання відпусток. На основі цього аналізу побудовано концептуальну модель бази даних у вигляді ER-діаграми, що відображає основні сутності, їх атрибути та зв'язки. Подальше проєктування логічної та фізичної моделей забезпечило створення нормалізованої OLTP-бази даних, приведеної до третьої нормальної форми та оснащеної необхідними обмеженнями цілісності й індексами.

У рамках курсової роботи реалізовано процес генерації та наповнення транзакційної бази значним обсягом тестових даних, що імітують реальні умови експлуатації системи. Це дозволило перевірити коректність структури бази даних, стійкість ETL-процесів та продуктивність аналітичної частини рішення при роботі з великими наборами інформації.

Ключовим етапом проєкту стала реалізація ETL-процесів за допомогою SQL Server Integration Services. Було побудовано повний цикл Extract → Transform → Load, у якому застосовано різні типи трансформацій, зокрема Derived Column, Lookup, Conditional Split, Data Conversion та Aggregate. Реалізовано механізми обробки помилок, розділення потоків даних на коректні та помилкові записи, а також логування результатів виконання ETL-пакетів. Це забезпечило високу якість даних, що завантажуються у сховище даних.

На основі підготовленого сховища даних побудовано OLAP-куб із використанням SQL Server Analysis Services. У кубі реалізовано набір вимірів та мір, обчислювані показники на основі MDX, елементи Time Intelligence, ранжування та KPI. Це дозволило виконувати багатовимірний аналіз кадрових даних у різних розрізах та часових періодах з високою швидкістю обробки запитів.

Для візуалізації аналітичної інформації розроблено комплекс звітів у SQL Server Reporting Services. Звіти охоплюють різні типи подання даних, зокрема табличні, матричні та графічні звіти, інтерактивну панель керування та звіти з можливістю деталізації (drill-down). Реалізовано параметризацію звітів із

використанням dropdown, multiselect, date range та cascading параметрів, що забезпечує гнучкий та зручний аналіз даних для користувача.

Проведений аналіз продуктивності підтвердив доцільність використання сховища даних і OLAP-куба для аналітичних задач. Порівняння виконання запитів у транзакційній базі та в OLAP-середовищі показало суттєвий виграш у швидкодії при використанні попередніх агрегацій та багатовимірної структури. Тестування та валідація системи засвідчили коректність обмежень цілісності, правильність роботи ETL-процесів і стабільність параметризованих аналітичних звітів.

Результати виконаної роботи демонструють практичну цінність побудованої системи як інструменту аналітичної підтримки управлінських рішень. Реалізоване рішення може бути використане для аналізу кадрових процесів, оцінки динаміки персоналу, контролю використання відпусток та планування ресурсів організації.

Подальший розвиток проєкту може передбачати розширення набору вимірів і мір OLAP-куба, зокрема додавання фінансових або організаційних показників. Доцільним є впровадження повноцінної підтримки повільно змінюваних вимірів (Slowly Changing Dimensions), автоматизація планування запусків ETL-процесів за розкладом, а також розширення системи логування та моніторингу. Крім того, можливе впровадження більш складних KPI та прогнозної аналітики, що дозволить підвищити аналітичну цінність системи та наблизити її до реальних BI-рішень, що використовуються у промислових умовах.

Додатки

SQL-скрипти

```
-- ============================================
-- OLTP база: HR_Variant7
```

```
-- ============================================

-- 1) Employee
CREATE TABLE dbo.Employee (
    EmployeeID        INT           NOT NULL,
    EmployeeCode      VARCHAR(20)   NOT NULL,
    LastName          NVARCHAR(50)  NOT NULL,
    FirstName         NVARCHAR(50)  NOT NULL,
    MiddleName        NVARCHAR(50)  NULL,
    BirthDate         DATE          NULL,
    Gender            CHAR(1)       NULL,
    HireDate          DATE          NOT NULL,
    TerminationDate   DATE          NULL,
    Email             VARCHAR(100)  NULL,
    Phone             VARCHAR(30)   NULL,
    Status            VARCHAR(20)   NOT NULL,
    CONSTRAINT PK_Employee PRIMARY KEY (EmployeeID),
    CONSTRAINT UQ_Employee_EmployeeCode UNIQUE (EmployeeCode),
    CONSTRAINT CK_Employee_Gender CHECK (Gender IS NULL OR Gender IN
('M','F')),
    CONSTRAINT CK_Employee_Dates CHECK (TerminationDate IS NULL OR
TerminationDate >= HireDate)
);

-- 2) Department
CREATE TABLE dbo.Department (
    DepartmentID      INT           NOT NULL,
    DepartmentName    NVARCHAR(100)  NOT NULL,
```

```
    ParentDepartmentID  INT           NULL,
    ManagerEmployeeID   INT           NULL,
    CONSTRAINT PK_Department PRIMARY KEY (DepartmentID),
    CONSTRAINT FK_Department_Parent
        FOREIGN KEY (ParentDepartmentID) REFERENCES
dbo.Department(DepartmentID),
    CONSTRAINT FK_Department_Manager
        FOREIGN KEY (ManagerEmployeeID) REFERENCES
dbo.Employee(EmployeeID)
);

-- 3) Position
CREATE TABLE dbo.Position (
    PositionID     INT           NOT NULL,
    PositionName   NVARCHAR(100)  NOT NULL,
    Grade          NVARCHAR(30)   NULL,
    BaseSalary     DECIMAL(12,2)  NOT NULL,
    IsActive       BIT           NOT NULL,
    CONSTRAINT PK_Position PRIMARY KEY (PositionID),
    CONSTRAINT CK_Position_BaseSalary CHECK (BaseSalary >= 0),
    CONSTRAINT CK_Position_IsActive CHECK (IsActive IN (0,1))
);

-- 4) VacationType
CREATE TABLE dbo.VacationType (
    VacationTypeID      INT           NOT NULL,
    TypeName            NVARCHAR(80)  NOT NULL,
    IsPaid              BIT           NOT NULL,
```

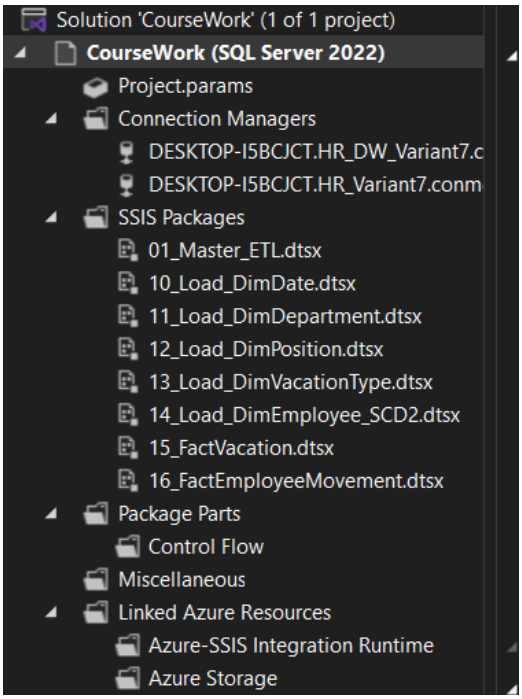
```
    DefaultDaysPerYear  INT          NULL,
    IsActive            BIT          NOT NULL,
    CONSTRAINT PK_VacationType PRIMARY KEY (VacationTypeID),
    CONSTRAINT CK_VacType_DefaultDays CHECK (DefaultDaysPerYear IS NULL OR DefaultDaysPerYear >= 0),
    CONSTRAINT CK_VacType_Flags CHECK (IsPaid IN (0,1) AND IsActive IN (0,1))
);

-- 5) Vacation
CREATE TABLE dbo.Vacation (
    VacationID      BIGINT       NOT NULL,
    EmployeeID      INT          NOT NULL,
    VacationTypeID  INT          NOT NULL,
    StartDate       DATE         NOT NULL,
    EndDate         DATE         NOT NULL,
    DaysCount       INT          NOT NULL,
    Status          VARCHAR(20)  NOT NULL,
    CreatedAt       DATETIME2(0) NOT NULL,
    ApprovedAt      DATETIME2(0) NULL,
    CONSTRAINT PK_Vacation PRIMARY KEY (VacationID),
    CONSTRAINT FK_Vacation_Employee
        FOREIGN KEY (EmployeeID) REFERENCES dbo.Employee(EmployeeID),
    CONSTRAINT FK_Vacation_VacationType
        FOREIGN KEY (VacationTypeID) REFERENCES dbo.VacationType(VacationTypeID),
    CONSTRAINT CK_Vacation_Dates CHECK (EndDate >= StartDate),
    CONSTRAINT CK_Vacation_DaysCount CHECK (DaysCount > 0),
```

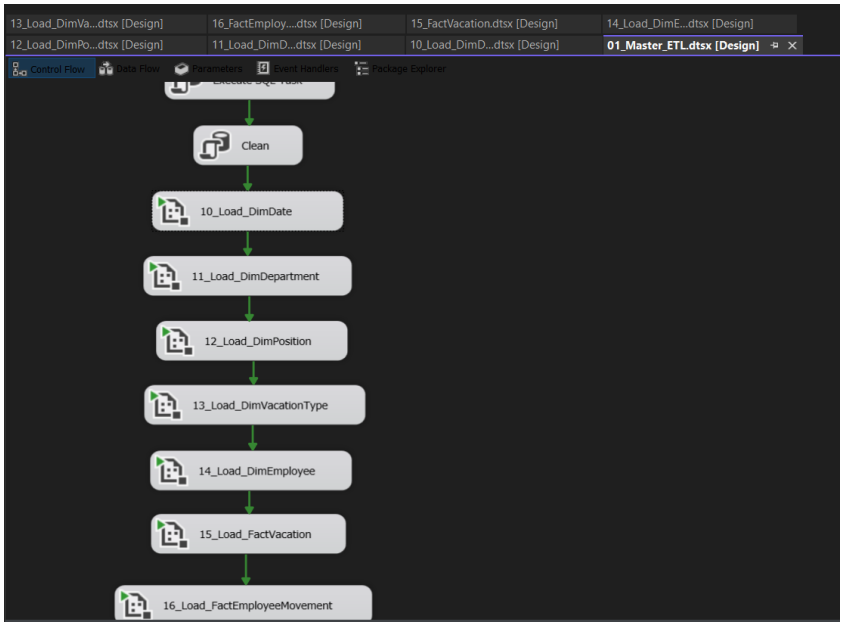
```
    CONSTRAINT CK_Vacation_ApprovedAt CHECK (ApprovedAt IS NULL OR ApprovedAt >= CreatedAt)
);

-- 6) EmployeePositionHistory
CREATE TABLE dbo.EmployeePositionHistory (
    PositionHistoryID  BIGINT        NOT NULL,
    EmployeeID         INT           NOT NULL,
    DepartmentID       INT           NOT NULL,
    PositionID         INT           NOT NULL,
    StartDate          DATE          NOT NULL,
    EndDate            DATE          NULL,
    Salary             DECIMAL(12,2)  NOT NULL,
    ChangeReason       NVARCHAR(200)  NULL,
    CONSTRAINT PK_EmployeePositionHistory PRIMARY KEY (PositionHistoryID),
    CONSTRAINT FK_EPH_Employee
        FOREIGN KEY (EmployeeID) REFERENCES dbo.Employee(EmployeeID),
    CONSTRAINT FK_EPH_Department
        FOREIGN KEY (DepartmentID) REFERENCES dbo.Department(DepartmentID),
    CONSTRAINT FK_EPH_Position
        FOREIGN KEY (PositionID) REFERENCES dbo.Position(PositionID),
    CONSTRAINT CK_EPH_Dates CHECK (EndDate IS NULL OR EndDate >= StartDate),
    CONSTRAINT CK_EPH_Salary CHECK (Salary >= 0)
);
```

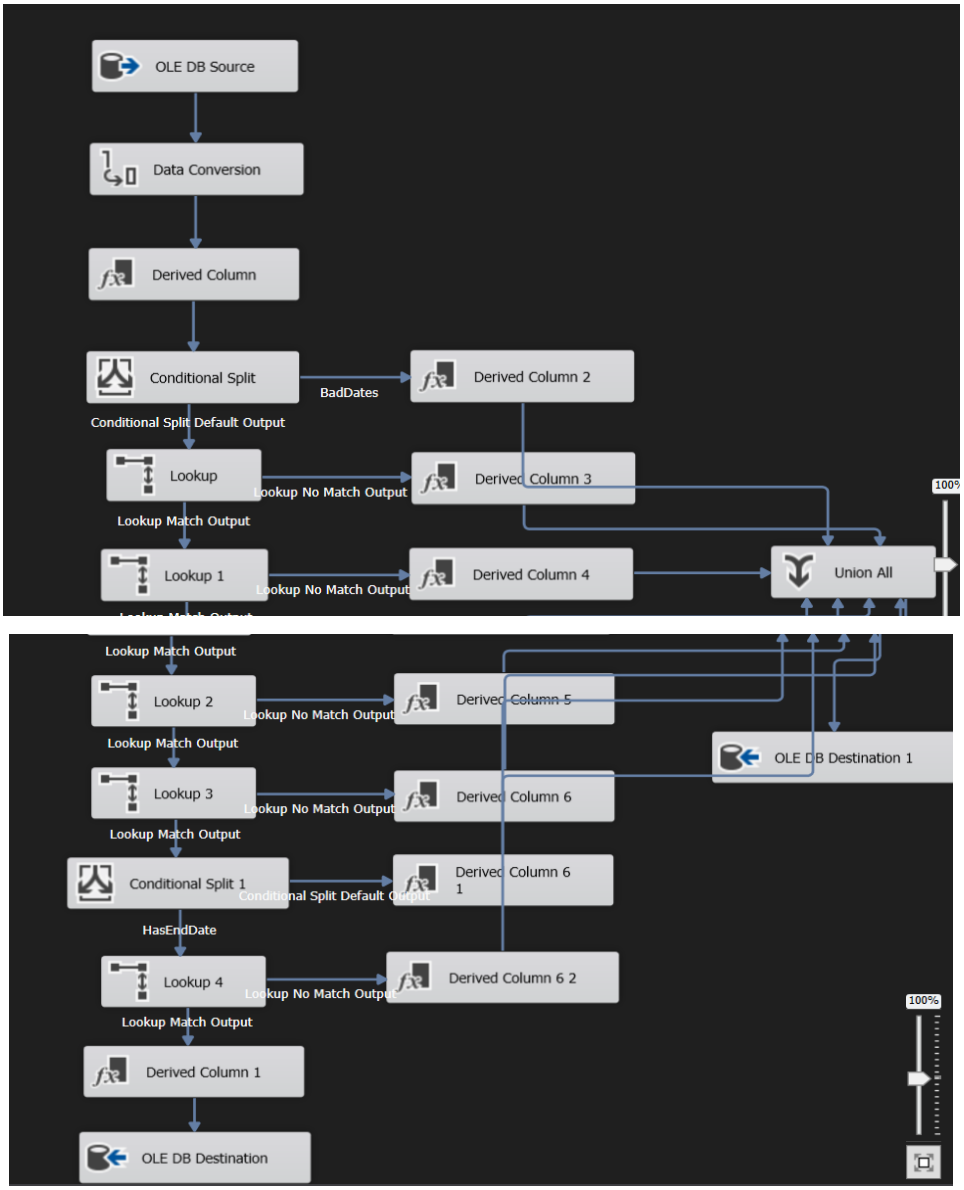
```
-- ============================================
-- Індекси для продуктивності (рекомендовано)
-- ============================================

CREATE NONCLUSTERED INDEX IX_Department_ParentDepartmentID
ON dbo.Department (ParentDepartmentID);

CREATE NONCLUSTERED INDEX IX_Department_ManagerEmployeeID
ON dbo.Department (ManagerEmployeeID);

CREATE NONCLUSTERED INDEX IX_Vacation_EmployeeID
ON dbo.Vacation (EmployeeID);

CREATE NONCLUSTERED INDEX IX_Vacation_VacationTypeID
ON dbo.Vacation (VacationTypeID);

CREATE NONCLUSTERED INDEX IX_Vacation_StartEndDate
ON dbo.Vacation (StartDate, EndDate);

CREATE NONCLUSTERED INDEX IX_EPH_EmployeeID
ON dbo.EmployeePositionHistory (EmployeeID);

CREATE NONCLUSTERED INDEX IX_EPH_DepartmentID
ON dbo.EmployeePositionHistory (DepartmentID);

CREATE NONCLUSTERED INDEX IX_EPH_PositionID
ON dbo.EmployeePositionHistory (PositionID);
```

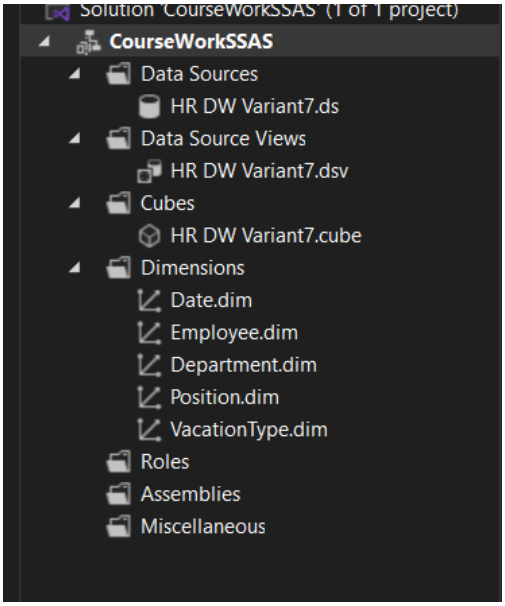
```
CREATE NONCLUSTERED INDEX IX_EPH_StartEndDate
ON dbo.EmployeePositionHistory (StartDate, EndDate);
```

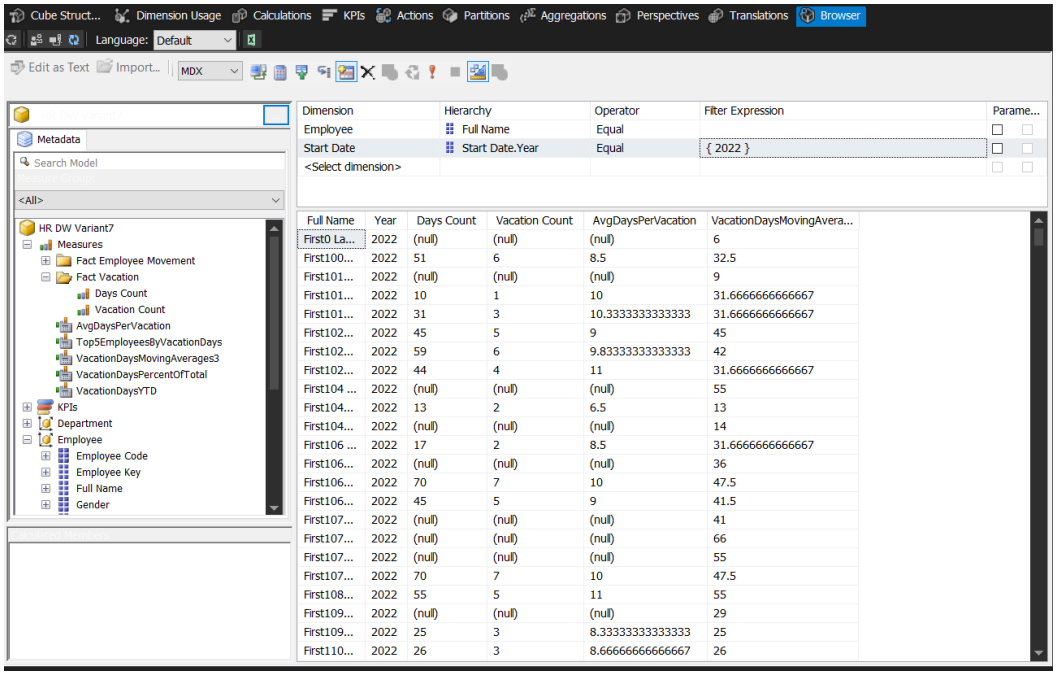
Скріншоти

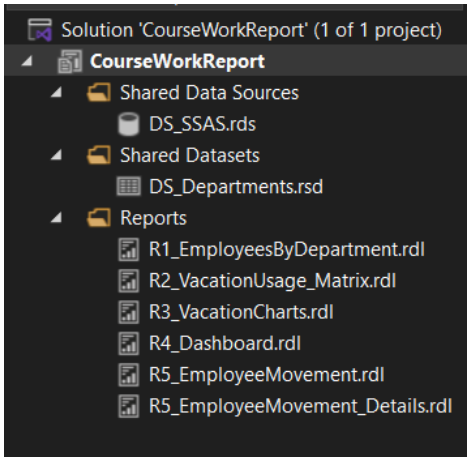





OLE DB Source
Data Conversion
Derived Column
Conditional Split
BadDates
Derived Column 2
Conditional Split Default Output
Lookup
Lookup No Match Output
Derived Column 3
Lookup Match Output
Lookup 1
Lookup No Match Output
Derived Column 4
Union All
100%
Lookup Match Output
Lookup 2
Lookup No Match Output
Derived Column 5
Lookup Match Output
OLE DB Destination 1
Lookup 3
Lookup No Match Output
Derived Column 6
Lookup Match Output
Conditional Split 1
Conditional Split Default Output
Derived Column 6 1
HasEndDate
Lookup 4
Lookup No Match Output
Derived Column 6 2
Lookup Match Output
Derived Column 1
OLE DB Destination
100%

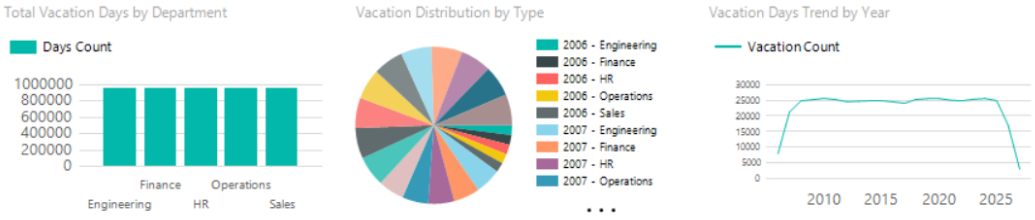

Solution 'CourseWorkSSAS' (1 of 1 project)
CourseWorkSSAS
Data Sources
HR DW Variant7.ds
Data Source Views
HR DW Variant7.dsv
Cubes
HR DW Variant7.cube
Dimensions
Date.dim
Employee.dim
Department.dim
Position.dim
VacationType.dim
Roles
Assemblies
Miscellaneous


Cube Struct...
Dimension Usage
Calculations
KPIs
Actions
Partitions
Aggregations
Perspectives
Translations
Browser
Language: Default
Edit as Text
Import...
MDX
Metadata
Search Model
<All>
HR DW Variant7
Measures
Fact Employee Movement
Fact Vacation
Days Count
Vacation Count
AvgDaysPerVacation
Top5EmployeesByVacationDays
VacationDaysMovingAverages3
VacationDaysPercentOfTotal
VacationDaysYTD
KPIs
Department
Employee
Employee Code
Employee Key
Full Name
Gender
Dimension
Hierarchy
Operator
Filter Expression
Parame...
Employee
Full Name
Equal
Start Date
Start Date.Year
Equal
{ 2022 }
<Select dimension>
Full Name
Year
Days Count
Vacation Count
AvgDaysPerVacation
VacationDaysMovingAvera...
First0 La... 2022 (null) (null) (null) 6
First100... 2022 51 6 8.5 32.5
First101... 2022 (null) (null) (null) 9
First101... 2022 10 1 10 31.6666666666667
First101... 2022 31 3 10.3333333333333 31.6666666666667
First102... 2022 45 5 9 45
First102... 2022 59 6 9.83333333333333 42
First102... 2022 44 4 11 31.6666666666667
First104 ... 2022 (null) (null) (null) 55
First104... 2022 13 2 6.5 13
First104... 2022 (null) (null) (null) 14
First106 ... 2022 17 2 8.5 31.6666666666667
First106... 2022 (null) (null) (null) 36
First106... 2022 70 7 10 47.5
First106... 2022 45 5 9 41.5
First107... 2022 (null) (null) (null) 41
First107... 2022 (null) (null) (null) 66
First107... 2022 (null) (null) (null) 55
First107... 2022 70 7 10 47.5
First108... 2022 55 5 11 55
First109... 2022 (null) (null) (null) 29
First109... 2022 25 3 8.33333333333333 25
First110... 2022 26 3 8.66666666666667 26


Solution 'CourseWorkReport' (1 of 1 project)
CourseWorkReport
Shared Data Sources
DS_SSAS.rds
Shared Datasets
DS_Departments.rsd
Reports
R1_EmployeesByDepartment.rdl
R2_VacationUsage_Matrix.rdl
R3_VacationCharts.rdl
R4_Dashboard.rdl
R5_EmployeeMovement.rdl
R5_EmployeeMovement_Details.rdl

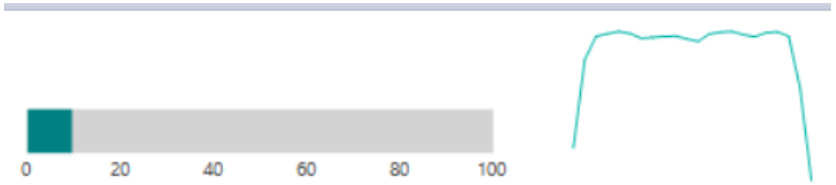
R3_VacationCharts

Department Name	Type Name	Year	Vacation Count	Days Count
Engineering	Annual Paid Leave	2006	323	2360
Engineering	Annual Paid Leave	2007	831	6315
Engineering	Annual Paid Leave	2008	1022	7615
Engineering	Annual Paid Leave	2009	993	7480
Engineering	Annual Paid Leave	2010	1035	7760
Engineering	Annual Paid Leave	2011	1010	7630
Engineering	Annual Paid Leave	2012	965	7255
Engineering	Annual Paid Leave	2013	981	7250
Engineering	Annual Paid Leave	2014	1018	7740
Engineering	Annual Paid Leave	2015	988	7340
Engineering	Annual Paid Leave	2016	979	7330
Engineering	Annual Paid Leave	2017	963	7250
Engineering	Annual Paid Leave	2018	1031	7650



R4_Dashboard

Department Name	Full Name	Year	Days Count	Vacation Count	Avg Days Per Vacation	Vacation Days Moving Averages3
Engineering	First6553 Last6553	2010	67	7	9.57142857142857	43
Finance	First6553 Last6553	2010	67	7	9.57142857142857	43
HR	First6553 Last6553	2010	67	7	9.57142857142857	43
Operations	First6553 Last6553	2010	67	7	9.57142857142857	43
Sales	First6553 Last6553	2010	67	7	9.57142857142857	43
Engineering	First6553 Last6553	2011	9	1	9	31.6666666666667
Finance	First6553 Last6553	2011	9	1	9	31.6666666666667
HR	First6553 Last6553	2011	9	1	9	31.6666666666667
Operations	First6553 Last6553	2011	9	1	9	31.6666666666667
Sales	First6553 Last6553	2011	9	1	9	31.6666666666667
Engineering	First6554 Last6554	2022	32	3	10.6666666666667	32
Finance	First6554 Last6554	2022	32	3	10.6666666666667	32
HR	First6554	2022	32	3	10.6666666666	32



Посилання

YouTube:

GitHub: https://github.com/Vovazast1/NVBD7