Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Інститут прикладної математики та фундаментальних наук

Кафедра прикладної математики

Курсова робота
з дисципліни «Надвеликі бази даних»

Виконав :
студент групи ПМ-42
Пилипчук О.О.

Перевірив :
доц. Любінський Б.Б.

_____
(дата) (підпис викладача)

Львів — 2026

Вступ

Актуальність теми. В умовах сучасного динамічного розвитку економіки ефективне управління людськими ресурсами стає одним із ключових факторів успіху будь-якого підприємства. Зростання масштабів організацій, збільшення штату працівників та ускладнення організаційних структур призводять до накопичення колосальних обсягів даних. Інформація про кадрові переміщення, історію посад, використання відпусток, нарахування заробітної плати та кваліфікацію персоналу формує так звані «великі дані», обробка яких стає викликом для традиційних систем обліку.

Для великих підприємств, де кількість працівників перевищує десятки тисяч, а історія ведеться роками, звичайні транзакційні бази даних (OLTP) перестають бути ефективними при формуванні аналітичної звітності. Виконання складних запитів для аналізу плинності кадрів або прогнозування витрат на відпустки може значно уповільнювати роботу системи. Саме тому виникає необхідність у впровадженні спеціалізованих систем бізнес-аналітики (Business Intelligence), які базуються на технологіях сховищ даних (Data Warehouse) та OLAP-кубів.

Використання стеку технологій Microsoft SQL Server дозволяє вирішити проблему розрізненості даних, забезпечити їх очищення та консолідацію, а також надати керівництву зручні інструменти для прийняття стратегічних рішень. Тому тема курсової роботи, присвячена розробці та аналізу надвеликої бази даних для предметної області «Облік кадрів», є безумовно актуальною.

Мета і завдання дослідження. Метою курсової роботи є розробка повнофункціональної системи керування надвеликою базою даних для предметної області «Облік кадрів» із реалізацією повного циклу обробки

даних: від проектування реляційної моделі до побудови аналітичної звітності з використанням технологій Microsoft SQL Server (SSIS, SSAS, SSRS).

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Провести детальний аналіз предметної області «Облік кадрів», виявити основні бізнес-процеси (найм, звільнення, переведення, надання відпусток) та сформулювати вимоги до системи.
2. Спроектувати концептуальну, логічну та фізичну моделі бази даних, забезпечивши нормалізацію до третьої нормальної форми (3НФ).
3. Згенерувати великий масив тестових даних (понад 500 000 записів), що імітують реальну діяльність підприємства за період 20 років, для перевірки продуктивності системи.
4. Розробити та реалізувати ETL-процеси (Extract, Transform, Load) за допомогою SQL Server Integration Services (SSIS) для перенесення даних у сховище типу Data Warehouse (схема «Зірка»).
5. Побудувати багатовимірний OLAP-куб засобами SQL Server Analysis Services (SSAS), налаштувати виміри, ієрархії та розрахункові міри для аналізу кадрових показників.
6. Розробити пакет інтерактивних аналітичних звітів у середовищі SQL Server Reporting Services (SSRS) та здійснити їх розгортання на сервері звітів із налаштуванням прав доступу.
7. Налаштувати політику безпеки та автоматичну розсилку звітності користувачам.

Об’єктом дослідження є процеси обліку та управління персоналом на великому підприємстві.

Предметом дослідження є методи та інструментальні засоби проектування надвеликих баз даних, побудови сховищ даних та систем бізнес-аналітики на платформі Microsoft SQL Server.

Методи дослідження. У роботі використано методи системного аналізу для дослідження предметної області, методи реляційної алгебри для проектування бази даних, методи багатовимірного аналізу даних для побудови OLAP-структур, а також методи візуалізації даних для створення звітності.

Практичне значення одержаних результатів полягає у створенні працездатного прототипу інформаційно-аналітичної системи, яка дозволяє автоматизувати рутинні операції відділу кадрів, скоротити час на формування звітності та підвищити якість управлінських рішень завдяки оперативному доступу до аналітики. Розроблена система демонструє можливості масштабування та обробки великих масивів даних, що підтверджується тестуванням на наборі даних обсягом понад півмільйона записів.

Структура роботи. Курсова робота складається зі вступу, чотирьох розділів, висновків, та списку використаних джерел.

Технічна реалізація інформаційно-аналітичної системи базується на екосистемі Microsoft SQL Server, яка є загальновизнаним стандартом у сфері корпоративної бізнес-аналітики. Для виконання поставлених завдань було використано такий інструментарій:

- **SQL Server Management Studio (SSMS)** — як основний засіб адміністрування реляційної бази даних та керування об’єктами сховища.
- **SQL Server Integration Services (SSIS)** — для забезпечення процесів ETL, що включають екстракцію, очищення та завантаження даних із транзакційної системи.
- **SQL Server Analysis Services (SSAS)** — для створення багатовимірних OLAP-структур, що забезпечують швидкий розрахунок агрегованих показників та аналіз даних у різних розрізах.
- **SQL Server Reporting Services (SSRS)** — для конструювання та публікації деталізованої звітності з можливістю інтерактивної взаємодії.

- **Visual Studio** — як єдине інтегроване середовище для розробки всіх компонентів

Впровадження розробленої системи дозволяє оптимізувати робочі процеси кадрової служби, мінімізувати ризики виникнення помилок ручного введення та забезпечити керівництво підприємства актуальною аналітикою. Створені звіти надають можливість оперативно моніторити кадрову динаміку, аналізувати використання відпусток та ефективність управління персоналом.

1.1 Аналіз предметної області

Детальний опис предметної області Предметною областю курсової роботи є інформаційно-аналітична система відділу кадрів. Система спроектована на основі дворівневої архітектури даних, що включає:

1. **Операційну базу даних (CW_Database)**: Використовується для поточної реєстрації кадрових подій. Основні сутності включають Employees (працівники), Positions (посади), Departments (відділи) та транзакційні таблиці Vacations і EmployeePositionHistory.
2. **Сховище даних (CW_DW)**: Використовується для аналітики. Сховище побудоване за схемою «Зірка» та містить дві основні таблиці фактів: FactVacations (для аналізу відпусток) та FactEmployeeHistory (для аналізу кадрових переміщень).

Така структура дозволяє розділити навантаження: операційні процеси (OLTP) не заважають формуванню складної аналітичної звітності (OLAP).

Основні бізнес-процеси (Автоматизація ETL) На основі розроблених пакетів Integration Services (SSIS), у системі автоматизовано такі ключові процеси обробки даних:

1. **Аудит та моніторинг виконання:** Перед початком завантаження система фіксує старт процесу в таблиці аудиту (SQL_Audit_Start), а після завершення

— оновлює статус (SQL_Audit_End). Це дозволяє адміністраторам контролювати успішність виконання щоденних оновлень.

2. **Синхронізація довідників (Loading Dimensions):** Процес, що забезпечує актуалізацію довідкової інформації (Departments, Positions, LeaveTypes). Використовується повне очищення (Truncate) та перезавантаження для забезпечення відповідності джерелу.
3. **Обробка історичних даних (Loading Facts):** Складний процес перенесення даних про історію переміщень (DF_Load_History) та відпустки (DF_Load_Vacations), який включає перевірку цілісності посилань на виміри.

Функціональні вимоги до системи Система розроблена для забезпечення наступних функціональних потреб:

- **Підтримка історії змін:** Забезпечення зберігання повної історії змін посад працівників у таблиці FactEmployeeHistory, що дозволяє будувати звіти "станом на дату".
- **Класифікація відпусток:** Можливість автоматичного розділення відпусток на категорії (наприклад, тривалі відпустки) для окремого аналізу.
- **Забезпечення якості даних:** Система не повинна допускати потрапляння у сховище записів про відпустки або переміщення, якщо для них відсутні відповідні записи у довідниках працівників чи посад.

Формулювання бізнес-правил та обмежень Під час реалізації ETL-процесів було імплементовано наступні бізнес-правила:

1. **Правило категоризації даних (Conditional Logic):** У потоці завантаження відпусток (DF_Load_Vacations) застосовано компонент **Conditional Split**. Система автоматично аналізує тривалість відпустки і, якщо вона перевищує встановлений ліміт, маркує такі записи як "Long Vacations" або направляє їх окремим потоком для спеціальної обробки. Всі інші записи проходять стандартний шлях через вихід за замовчуванням.

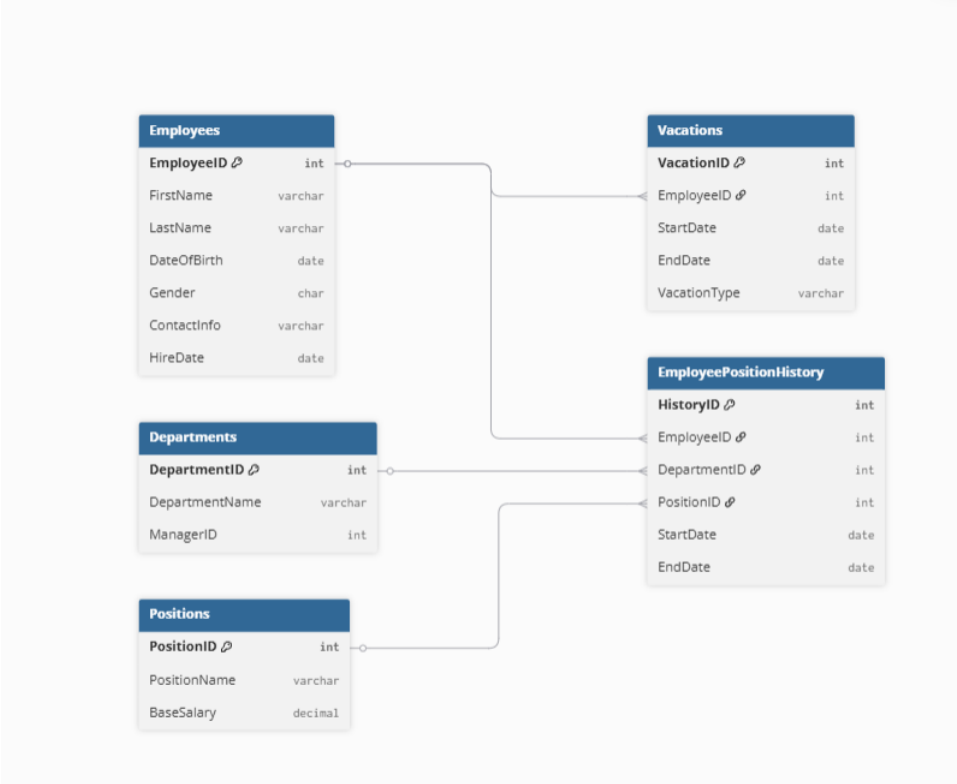
2. **Правило цілісності зв'язків (Lookup Logic):** При завантаженні історії переміщень (DF_Load_History) використовується каскад компонентів **Lookup**. Завантаження факту можливе лише за умови успішного знаходження сурогатних ключів для Працівника, Відділу та Посади. Це гарантує, що у таблицю фактів не потраплять "сироти" (записи без прив'язки до вимірів).
3. **Правило уніфікації типів даних:** Оскільки джерело та призначення можуть мати різні кодування (Unicode/Non-Unicode), застосовано компонент **Data Conversion** для явного приведення текстових полів до формату DT_WSTR, що відповідає стандартам сховища даних.
4. **Правило консолідації:** Використання компонента **Union All** дозволяє об'єднати різні потоки даних (наприклад, стандартні та тривалі відпустки) в єдиний потік перед фінальним записом у базу даних.

1.2 Концептуальне проектування

На основі аналізу предметної області та функціональних вимог було розроблено концептуальну модель бази даних. Для графічного представлення моделі обрано нотацію **Crow's Foot** («Лапка ворони»), яка є стандартом при проектуванні реляційних баз даних та підтримується середовищем MS SQL Server Management Studio.

1. Побудова ER-діаграми

ER-діаграма (Entity-Relationship Diagram) відображає статичну структуру системи. Центральним елементом моделі є сутність Employees, навколо якої будуються зв'язки з довідниками (Departments, Positions) та транзакційними сутностями (Vacations, EmployeePositionHistory).



2. Опис сутностей та їх атрибутів

На основі фізичної реалізації виділено такі сутності:

1. **Employees (Працівники)**
   - *Тип:* Стрижнева (Strong) сутність.
   - *Опис:* Зберігає персональні дані співробітників.
   - *Ключові атрибути:*
     - EmployeeID (PK) — унікальний ідентифікатор (табельний номер).
     - FirstName, LastName — ім'я та прізвище.
     - BirthDate — дата народження (для вікового аналізу).
     - Gender — стать.
     - HireDate — дата першого найму.
2. **Departments (Відділи)**
   - *Тип:* Довідкова сутність.
   - *Опис:* Містить перелік структурних підрозділів компанії.
   - *Ключові атрибути:*
     - DepartmentID (PK) — ідентифікатор відділу.

- DepartmentName — назва підрозділу.

3. **Positions (Посади)**

- *Опис:* Штатний розклад із зазначенням базових ставок.
- *Ключові атрибути:*

- PositionID (PK) — код посади.
- PositionTitle — назва посади.
- BaseSalary — базовий оклад.

4. **Vacations (Відпустки)**

- *Опис:* Реєстр періодів відсутності працівників на робочому місці.
- *Ключові атрибути:*

- VacationID (PK) — унікальний номер запису.
- EmployeeID (FK) — посилання на працівника.
- StartDate, EndDate — період відпустки.

5. **EmployeePositionHistory (Історія призначень)**

- *Опис:* Забезпечує реалізацію вимоги щодо зберігання історії переміщень (зв'язок "багато-до-багатьох" розгорнутий у часі).
- *Ключові атрибути:*

- HistoryID (PK) — ідентифікатор запису історії.
- EmployeeID, DepartmentID, PositionID (FK) — зовнішні ключі.
- StartDate — дата початку роботи на посаді.
- EndDate — дата завершення (або NULL, якщо посада актуальна).

3. Визначення зв'язків та опис кардинальності

Всі зв'язки у системі є ідентифікуючими або неідентифікуючими зв'язками типу **«Один-до-багатьох» (1:M)**, що забезпечує цілісність даних:

1. **Employees — Vacations (1:M)**

- *Кардинальність:* Один (1) працівник може мати нуль, одну або багато (M) відпусток. Кожен запис про відпустку обов'язково належить лише одному працівнику.
- *Правило бізнесу:* Працівник може йти у відпустку багаторазово протягом своєї кар'єри.

2. **Employees — EmployeePositionHistory (1:M)**
- *Кардинальність:* Один (1) працівник має одну або багато (M) записів в історії (мінімум один запис про прийом на роботу).
- *Правило бізнесу:* Цей зв'язок дозволяє відстежувати кар'єрний ріст працівника.

3. **Departments — EmployeePositionHistory (1:M)**
- *Кардинальність:* Один (1) відділ може фігурувати у багатьох (M) записах історії призначень різних працівників.
- *Правило бізнесу:* Відділ складається з багатьох працівників, склад яких змінюється у часі.

4. **Positions — EmployeePositionHistory (1:M)**
- *Кардинальність:* Одна (1) посада призначається багатьом (M) працівникам у різні періоди часу.

1.3 Логічне проектування

Логічне проектування бази даних виконано з метою забезпечення цілісності даних, усунення надлишковості та оптимізації швидкодії системи.

Нормалізація бази даних

У процесі проектування схема бази даних була приведена до **Третьої нормальної форми (3НФ)**, що є необхідною умовою для транзакційних систем (OLTP).

1. **Перша нормальна форма (1НФ):**
   - Усі таблиці містять лише атомарні (неподільні) значення. Наприклад, у таблиці Employees адреса або контактна інформація зберігаються в окремих полях, а не списком.
   - Відсутні групи атрибутів, що повторюються. Історія посад винесена в окрему таблицю EmployeePositionHistory, а не зберігається у вигляді колонок Position1, Position2 у таблиці працівника.
   - Усі записи унікально ідентифікуються первинним ключем.
2. **Друга нормальна форма (2НФ):**
   - База даних знаходиться в 1НФ.
   - Усі неключові атрибути повністю залежать від первинного ключа. Оскільки в таблицях використовуються прості (не складені) сурогатні ключі (EmployeeID, DepartmentID), проблема часткової залежності відсутя. Наприклад, BaseSalary у таблиці Positions залежить лише від PositionID.
3. **Третя нормальна форма (3НФ):**
   - База даних знаходиться в 2НФ.
   - Відсутні транзитивні залежності між неключовими атрибутами.
   - *Приклад:* У таблиці EmployeePositionHistory ми зберігаємо лише DepartmentID та PositionID. Ми не зберігаємо там DepartmentName або BaseSalary. Якщо зміниться назва відділу, нам потрібно оновити її лише в одному місці (таблиця Departments), і ці зміни автоматично відобразяться для всіх працівників. Це усуває аномалії оновлення.

Визначення первинних та зовнішніх ключів

Для забезпечення посилальної цілісності (Referential Integrity) у схемі визначено такі ключі:

Таблиця	**Первинний ключ (PK)**	**Зовнішні ключі (FK)**	**Опис зв’язку**

Employees	EmployeeID	—	Унікальний ідентифікатор працівника.
Departments	DepartmentID	ManagerID (FK -> Employees)	Керівник є також працівником компанії.
Positions	PositionID	—	Унікальний код посади.
Vacations	VacationID	EmployeeID (FK -> Employees)	Зв’язує відпустку з конкретним працівником.
History	HistoryID	EmployeeID, DepartmentID, PositionID	Зв’язує запис історії з працівником, відділом та посадою.

Використання обмежень FOREIGN KEY гарантує, що неможливо видалити відділ, у якому працюють люди, або додати відпустку неіснуючому працівнику.

Проектування індексів для оптимізації запитів

Оскільки система повинна працювати з великими обсягами даних (понад 500 000 записів), розроблено стратегію індексації:

1. **Кластерні індекси (Clustered Indexes):**
   - Автоматично створені для всіх стовпців **Primary Key**. Це фізично впорядковує дані на диску за ID, що пришвидшує пошук конкретного запису.
2. **Некластерні індекси (Non-Clustered Indexes):**
   - Створені для стовпців **Foreign Key** (EmployeeID у таблицях Vacations та History). Це критично важливо для прискорення операцій JOIN, які використовуються у звітах.
   - *Індекс для пошуку:* Створено індекс по полю FullName у таблиці Employees, оскільки пошук працівників за прізвищем та іменем є найчастішою операцією HR-менеджера.
   - *Індекс для фільтрації:* Створено індекс по полю StartDate у таблиці Vacations для швидкого вибору відпусток за певний період (рік/місяць).

1.4 Фізичне проектування

На цьому етапі було реалізовано фізичну модель бази даних засобами мови SQL. Нижче наведено скрипти створення основних таблиць, обмежень цілісності та збережених процедур.

1. SQL-скрипти створення таблиць

Для створення таблиць використано команди CREATE TABLE. Типи даних підібрано з урахуванням специфіки інформації (наприклад, NVARCHAR для текстових полів, щоб підтримувати кирилицю).

```sql
USE CW_Database;
GO

-- 1. Створення таблиці Посад (Positions)
CREATE TABLE Positions (
    PositionID INT IDENTITY(1,1) NOT NULL,
    PositionName NVARCHAR(100) NOT NULL,
    BaseSalary DECIMAL(10, 2) NOT NULL CHECK (BaseSalary > 0),
    CONSTRAINT PK_Positions PRIMARY KEY CLUSTERED (PositionID)
);
GO

-- 2. Створення таблиці Відділів (Departments)
CREATE TABLE Departments (
    DepartmentID INT IDENTITY(1,1) NOT NULL,
    DepartmentName NVARCHAR(100) NOT NULL,
    ManagerID INT NULL,
    CONSTRAINT PK_Departments PRIMARY KEY CLUSTERED (DepartmentID)
);
GO

-- 3. Створення таблиці Працівників (Employees)
CREATE TABLE Employees (
    EmployeeID INT IDENTITY(1,1) NOT NULL,
    FirstName NVARCHAR(50) NOT NULL,
    LastName NVARCHAR(50) NOT NULL,
    DateOfBirth DATE NOT NULL,
    Gender CHAR(1) CHECK (Gender IN ('M', 'F')),
    ContactInfo NVARCHAR(200),
    HireDate DATE NOT NULL,
    CONSTRAINT PK_Employees PRIMARY KEY CLUSTERED (EmployeeID)
);
GO
```

```sql
-- 4. Створення таблиці Історії Посад (EmployeePositionHistory)
CREATE TABLE EmployeePositionHistory (
    HistoryID INT IDENTITY(1,1) NOT NULL,
    EmployeeID INT NOT NULL,
    DepartmentID INT NOT NULL,
    PositionID INT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NULL, -- NULL означає, що це поточна посада

    CONSTRAINT PK_EmployeePositionHistory PRIMARY KEY CLUSTERED (HistoryID),

    CONSTRAINT FK_History_Employee FOREIGN KEY (EmployeeID)
        REFERENCES Employees(EmployeeID) ON DELETE CASCADE,
    CONSTRAINT FK_History_Department FOREIGN KEY (DepartmentID)
        REFERENCES Departments(DepartmentID),
    CONSTRAINT FK_History_Position FOREIGN KEY (PositionID)
        REFERENCES Positions(PositionID),

    CONSTRAINT CK_History_Dates CHECK (EndDate IS NULL OR EndDate >= StartDate)
);
GO

-- 5. Створення таблиці Відпусток (Vacations)
CREATE TABLE Vacations (
    VacationID INT IDENTITY(1,1) NOT NULL,
    EmployeeID INT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NOT NULL,
    VacationType NVARCHAR(50) NOT NULL CHECK (VacationType IN (N'Планова', N'Лікарняний', N'За свій рахунок')),
```

```
    VacationType NVARCHAR(50) NOT NULL CHECK (VacationType IN (N'Планова', N'Лікарняний', N'За свій рахунок')),

    CONSTRAINT PK_Vacations PRIMARY KEY CLUSTERED (VacationID),

    CONSTRAINT FK_Vacations_Employee FOREIGN KEY (EmployeeID)
        REFERENCES Employees(EmployeeID) ON DELETE CASCADE,

    CONSTRAINT CK_Vacation_Dates CHECK (EndDate >= StartDate)
);
GO

-- Додавання зовнішнього ключа для Керівника відділу
ALTER TABLE Departments
ADD CONSTRAINT FK_Departments_Manager FOREIGN KEY (ManagerID)
    REFERENCES Employees(EmployeeID);
GO


CREATE NONCLUSTERED INDEX IX_Employees_LastName ON Employees(LastName);
CREATE NONCLUSTERED INDEX IX_History_EmployeeID ON EmployeePositionHistory(EmployeeID);
CREATE NONCLUSTERED INDEX IX_History_DepartmentID ON EmployeePositionHistory(DepartmentID);
CREATE NONCLUSTERED INDEX IX_History_PositionID ON EmployeePositionHistory(PositionID);
CREATE NONCLUSTERED INDEX IX_History_EndDate ON EmployeePositionHistory(EndDate) WHERE EndDate IS NULL;
CREATE NONCLUSTERED INDEX IX_Vacations_StartDate ON Vacations(StartDate);
GO
```

2. Реалізація обмежень цілісності (Constraints, CHECK,)

Для забезпечення логічної коректності та узгодженості даних у базі було реалізовано декларативні обмеження цілісності двох типів: FOREIGN KEY та CHECK.**3. Створення тригерів (Triggers)**

Реалізовано тригер для автоматичної перевірки бізнес-правил. Наприклад, тригер, що забороняє видалення відділу, якщо в ньому ще працюють люди (згідно з історією).

```
USE [CW_Database];
GO

CREATE TRIGGER [dbo].[trg_PreventDepartmentDelete]
ON [dbo].[Departments]
INSTEAD OF DELETE
AS
BEGIN
    SET NOCOUNT ON;

    -- Перевірка: чи є записи в історії по цьому відділу?
    IF EXISTS (SELECT 1 FROM deleted d
               JOIN dbo.EmployeePositionHistory h ON d.DepartmentID = h.DepartmentID)
    BEGIN
        -- Якщо є — видавати помилку і скасовувати видалення
        RAISERROR ('Неможливо видалити відділ, оскільки в ньому є історія роботи працівників.', 16, 1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        -- Якщо історії немає — дозволити видалення
        DELETE FROM [dbo].[Departments]
        WHERE DepartmentID IN (SELECT DepartmentID FROM deleted);
    END
END;
GO
```

4. Створення збережених процедур

Розроблено процедуру для спрощення додавання нових працівників.

```
USE [CW_Database];
GO

CREATE PROCEDURE [dbo].[usp_AddNewEmployee]
    @FirstName NVARCHAR(50),
    @LastName NVARCHAR(50),
    @BirthDate DATE,
    @Gender NCHAR(1),
    @HireDate DATE
AS
BEGIN
    SET NOCOUNT ON;

    INSERT INTO [dbo].[Employees] ([FirstName], [LastName], [DateOfBirth], [Gender], [HireDate])
    VALUES (@FirstName, @LastName, @BirthDate, @Gender, @HireDate);

    SELECT SCOPE_IDENTITY() AS NewEmployeeID;
END;
GO
```

ЕТАП 2 ГЕНЕРАЦІЯ ТА НАПОВНЕННЯ ДАНИХ

2.1. Підготовка до генерації даних

Для тестування продуктивності сховища даних та перевірки коректності роботи ETL-процесів необхідно наповнити базу даних великим обсягом інформації ("Big Data").

Вибір інструменту генерації В якості основного інструменту обрано **Redgate SQL Data Generator**. Цей вибір обґрунтований наступними факторами, підтвердженими в ході налаштування проекту:

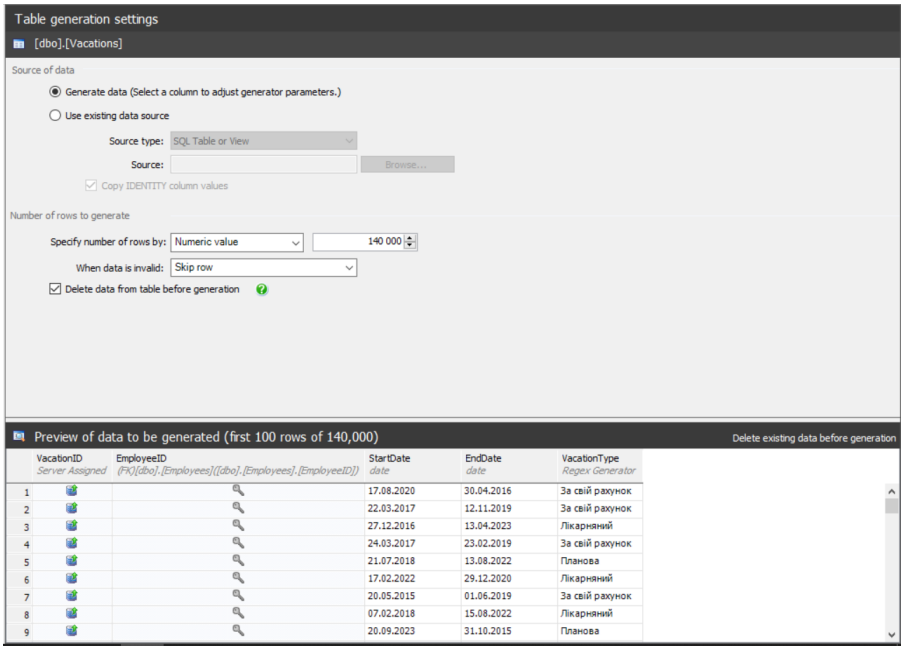
1. **Автоматична підтримка цілісності (Referential Integrity):** Інструмент автоматично зчитує схему бази даних та враховує зв'язки FOREIGN KEY. Наприклад, при генерації таблиці Departments, поле ManagerID автоматично заповнюється існуючими EmployeeID з таблиці працівників.
2. **Гнучкість налаштувань (Custom Generators):** Можливість використання регулярних виразів (Regex) для генерації специфічних доменних даних. Для стовпця VacationType було створено кастомний генератор, що випадковим чином обирає значення зі списку: *"Планова", "Лікарняний", "За свій рахунок"*.
3. **Висока швидкість:** Можливість згенерувати сотні тисяч записів за лічені хвилини без написання складних SQL-циклів.

Аналіз вимог до реалістичності даних Щоб згенеровані дані виглядали достовірно, було налаштовано наступні параметри генераторів:

- **Персональні дані:** Для таблиці Employees використано генератор "Person", що створює реалістичні імена (FirstName, LastName) та дати народження в діапазоні 1970–2000 років.
- **Фінансові дані:** Для таблиці Positions налаштовано генератор decimal, що формує базові оклади (BaseSalary) у реалістичному діапазоні від 12 000 до 47 000
- **Часові ряди:** Для відпусток забезпечено логіку, де EndDate завжди пізніше за StartDate.

Визначення стратегії розподілу даних На основі налаштувань проекту генерації затверджено наступну стратегію розподілу обсягів даних:

1. **Довідкові дані (Dimensions):**
   - **Departments:** 10 записів (основні відділи: Security, Legal, Logistics та ін.).
   - **Positions:** 15 записів (штатний розклад: від Recruiter до Director).
   - **Employees: 10 000 записів**. Такий обсяг дозволяє імітувати велике підприємство та створює достатнє навантаження для тестування швидкодії запитів.
2. **Транзакційні дані (Facts):**
   - **Vacations: 140 000 записів**. Це забезпечує в середньому по 14 записів про відпустки на кожного працівника за весь період історії, що дозволяє будувати змістовні аналітичні звіти по роках.
   - **EmployeePositionHistory:** Наповнюється пропорційно до кількості працівників для моделювання кар'єрного росту.



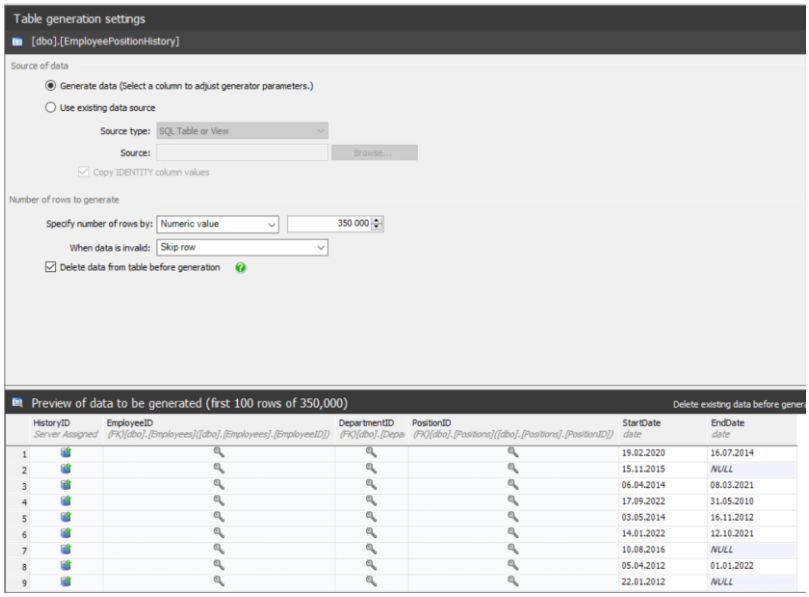
2.2 Реалізація генерації великих обсягів даних

Для забезпечення навантажувального тестування сховища даних було виконано генерацію масивів інформації, що імітують діяльність великого підприємства протягом тривалого періоду.

Генерація таблиць фактів (Transaction Data)

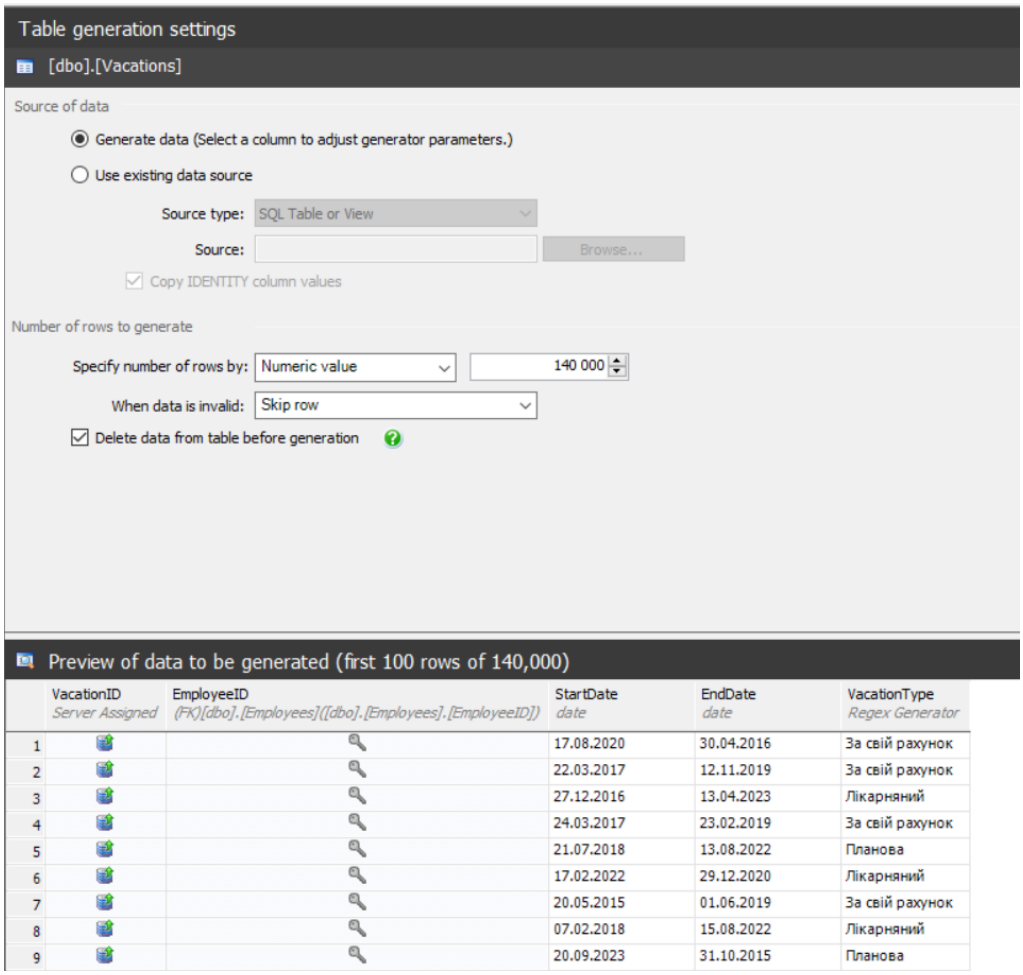
Основний акцент було зроблено на наповненні таблиць фактів, оскільки саме вони створюють основне навантаження на ETL-процеси. Загальний обсяг згенерованих транзакційних даних складає **490 000 записів** (наближено до цільового показника 500 000):

- **EmployeePositionHistory (Історія переміщень):** Згенеровано **350 000 записів**.
  - *Мета:* Цей масив даних дозволяє протестувати алгоритми обробки повільно змінюваних вимірів (SCD Type 2). При наявності 10 000 працівників це означає, що в середньому кожен співробітник має 35 записів в історії кар'єри, що забезпечує глибоку історичну перспективу для аналізу.



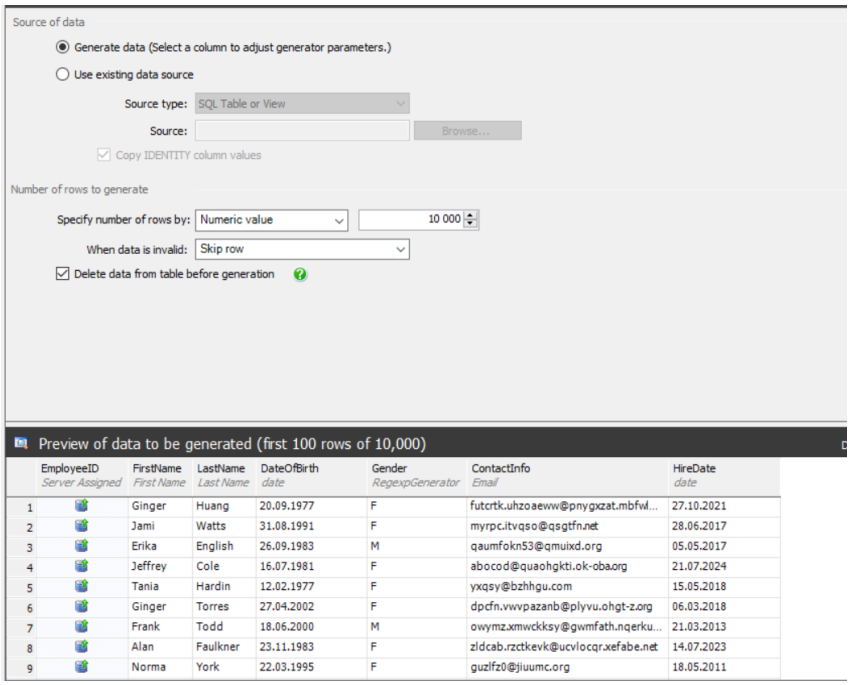
Vacations (Відпустки): Згенеровано **140 000 записів**.

Мета: Створено дані про відсутність персоналу, що покривають різні типи відпусток (планові, лікарняні, за свій рахунок) з коректним розподілом у часі.



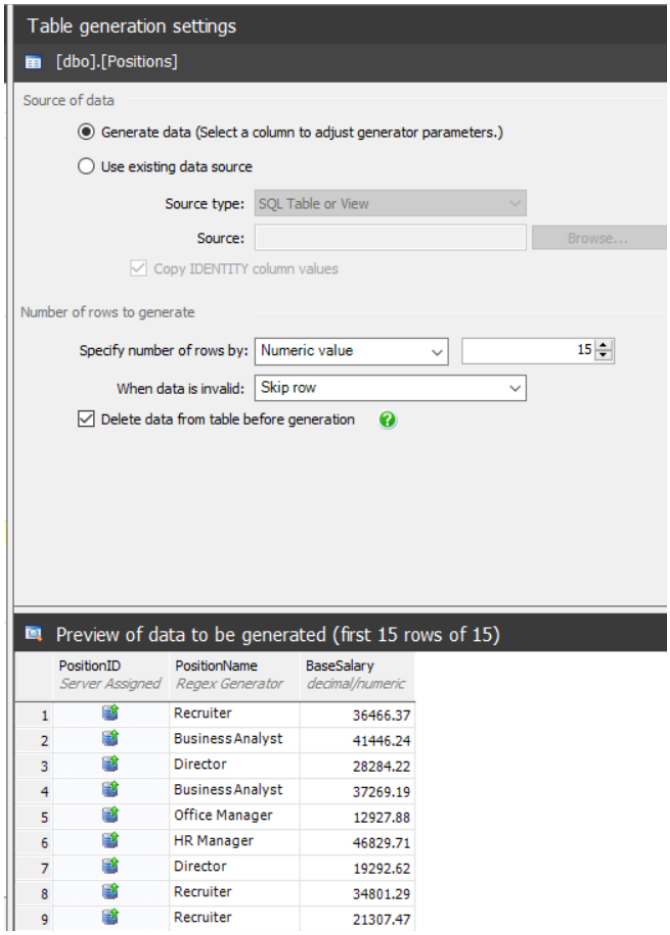
Генерація довідкових даних (Master Data)

- **Employees (Працівники):** Таблицю наповнено **10 000 унікальних профілів**.
  - Генератор забезпечив створення реалістичних українських та англійських імен, коректних дат народження (вік працівників від 18 до 65 років) та унікальних контактних даних.

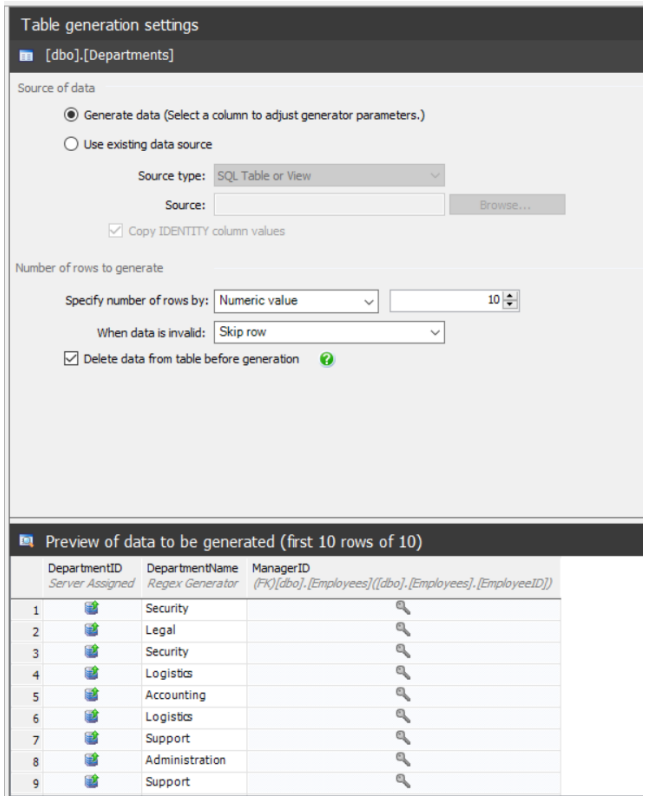


	EmployeeID *Server Assigned*	FirstName *First Name*	LastName *Last Name*	DateOfBirth *date*	Gender *RegexpGenerator*	ContactInfo *Email*	HireDate *date*
1		Ginger	Huang	20.09.1977	F	futcrtk.uhzoaeww@pnygxzat.mbfwl...	27.10.2021
2		Jami	Watts	31.08.1991	F	myrpc.itvqso@qsgtfn.net	28.06.2017
3		Erika	English	26.09.1983	M	qaumfokn53@qmuixd.org	05.05.2017
4		Jeffrey	Cole	16.07.1981	F	abocod@quaohgkti.ok-oba.org	21.07.2024
5		Tania	Hardin	12.02.1977	F	yxqsy@bzhhgu.com	15.05.2018
6		Ginger	Torres	27.04.2002	F	dpcfn.vwvpazanb@plyvu.ohgt-z.org	06.03.2018
7		Frank	Todd	18.06.2000	M	owymz.xmwckksy@gwmfath.nqerku...	21.03.2013
8		Alan	Faulkner	23.11.1983	F	zldcab.rzctkevk@ucvlocqr.xefabe.net	14.07.2023
9		Norma	York	22.03.1995	F	guzlfz0@jiuumc.org	18.05.2011

- **Dictionaries (Positions, Departments):**
  - Створено **15 ключових посад** (від Recruiter до Director) із реалістичними ставками заробітної плати.
  - Створено **10 структурних підрозділів** (Security, Legal, Accounting тощо) для розподілу працівників по відділах.



	PositionID *Server Assigned*	PositionName *Regex Generator*	BaseSalary *decimal/numeric*
1		Recruiter	36466.37
2		Business Analyst	41446.24
3		Director	28284.22
4		Business Analyst	37269.19
5		Office Manager	12927.88
6		HR Manager	46829.71
7		Director	19292.62
8		Recruiter	34801.29
9		Recruiter	21307.47



	DepartmentID *Server Assigned*	DepartmentName *Regex Generator*	ManagerID *(FK)[dbo].[Employees]([dbo].[Employees].[EmployeeID])*
1		Security	
2		Legal	
3		Security	
4		Logistics	
5		Accounting	
6		Logistics	
7		Support	
8		Administration	
9		Support	

3. Забезпечення реалістичності та бізнес-логіки

Використання **Redgate SQL Data Generator** дозволило дотриматися суворих правил бізнес-логіки під час генерації:

1. **Цілісність посилань (Referential Integrity):** Усі 350 000 записів в таблиці історії посилаються виключно на існуючі EmployeeID (з діапазону 1-10000) та DepartmentID. "Биті" посилання відсутні.
2. **Часова логіка (Temporal Consistency):**
   - У таблиці відпусток дата завершення (EndDate) завжди більша за дату початку (StartDate).
   - Дати найму (HireDate) узгоджені з віком працівника (ніхто не найнятий у віці 5 років).
3. **Розподіл даних:** Використано випадковий розподіл (Random Distribution) для типів відпусток та посад, що дозволяє уникнути статистичних перекосів у звітах.

2.3 Налаштування генератора даних

Для забезпечення максимальної реалістичності тестових даних було виконано детальну конфігурацію генераторів для кожної таблиці окремо. Налаштування виконувались у середовищі **Redgate SQL Data Generator** з використанням регулярних виразів та вбудованих алгоритмів розподілу.

1. Конфігурація правил генерації для кожної таблиці

Для кожної сутності було підібрано специфічні типи генераторів:

- **Employees (Персональні дані):**
  - Для полів FirstName та LastName використано генератор **Person**, налаштований на створення імен у форматі "Ім'я Прізвище".
  - Для поля DateOfBirth встановлено діапазон генерації від **01.01.1970 до 31.12.2005**, що забезпечує віковий склад працівників від 18 до 55 років.

- Для ContactInfo (Email) використано шаблон регулярного виразу, що генерує унікальні корпоративні адреси.

- **Vacations (Категоріальні дані):**
- Для поля VacationType використано **Regex Generator** з переліком допустимих значень українською мовою: *("Планова"|"Лікарняний"|"За свій рахунок")*. Це дозволяє уникнути появи випадкового тексту ("Lorem Ipsum") у звітах.
- Для дат налаштовано залежність: EndDate = StartDate + (5...24 days).
- **Positions (Фінансові дані):**
- Для поля BaseSalary використано тип генератора decimal. Встановлено діапазон від **12 000 до 47 000**, що відповідає ринковим реаліям для обраних посад.

2. Забезпечення зв'язності даних (Data Connectivity)

Інструмент автоматично проаналізував схему бази даних та виявив зв'язки FOREIGN KEY.

- У налаштуваннях таблиці EmployeePositionHistory поле EmployeeID автоматично налаштовано на вибірку існуючих ключів з таблиці Employees. Це гарантує, що історія генерується тільки для реально існуючих працівників.
- Аналогічно налаштовано зв'язок ManagerID у таблиці відділів, що дозволяє уникнути порушення посилальної цілісності.

3. Генерація даних з урахуванням часових періодів

Однією з головних вимог була наявність історичних даних за тривалий період (3-5 років) для аналізу трендів.

- У генераторі для поля StartDate таблиці EmployeePositionHistory було задано часове вікно **з 2015 по 2024 рік**.
- Це забезпечило рівномірний розподіл подій (прийом на роботу, звільнення, відпустки) по всьому часовому інтервалу, що дозволить будувати звіти типу "Year-over-Year" (порівняння рік до року).

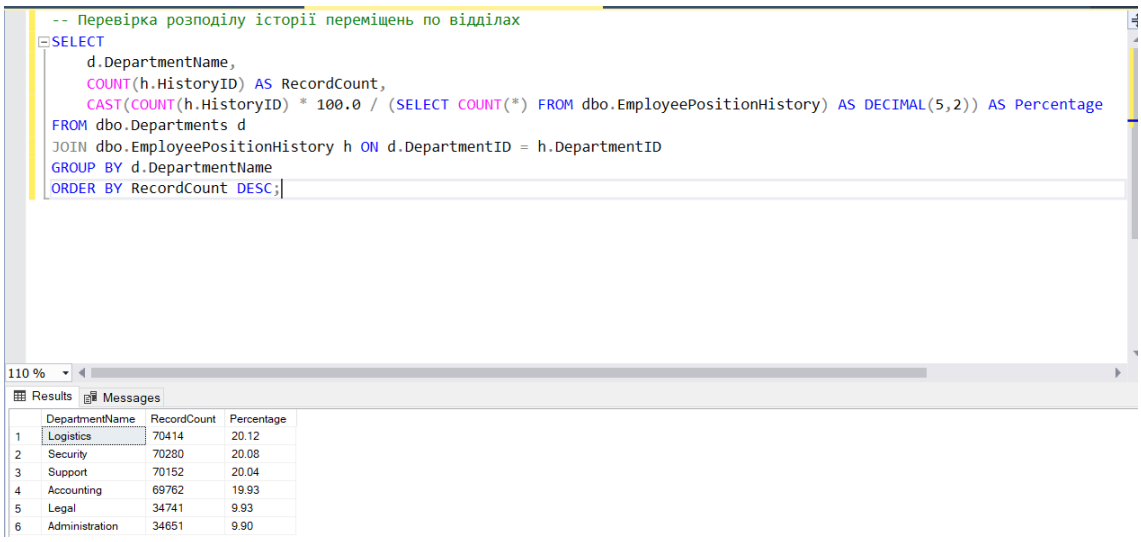
2.4 Верифікація даних

Після завершення процесу генерації було проведено комплексний аудит отриманого масиву даних. Метою верифікації є підтвердження того, що синтетичні дані придатні для використання в аналітичній системі та не призведуть до помилок під час виконання ETL-процесів.

1. Перевірка структурної цілісності та розподілу

Простого підрахунку рядків недостатньо, тому було виконано перевірку статистичного розподілу даних. За допомогою SQL-запитів з групуванням (GROUP BY) перевірено, чи рівномірно розподілені працівники по відділах та чи коректно заповнена історія.

SQL-скрипт аналізу розподілу навантаження:



```sql
-- Перевірка розподілу історії переміщень по відділах
SELECT
    d.DepartmentName,
    COUNT(h.HistoryID) AS RecordCount,
    CAST(COUNT(h.HistoryID) * 100.0 / (SELECT COUNT(*) FROM dbo.EmployeePositionHistory) AS DECIMAL(5,2)) AS Percentage
FROM dbo.Departments d
JOIN dbo.EmployeePositionHistory h ON d.DepartmentID = h.DepartmentID
GROUP BY d.DepartmentName
ORDER BY RecordCount DESC;
```

110 %

Results | Messages

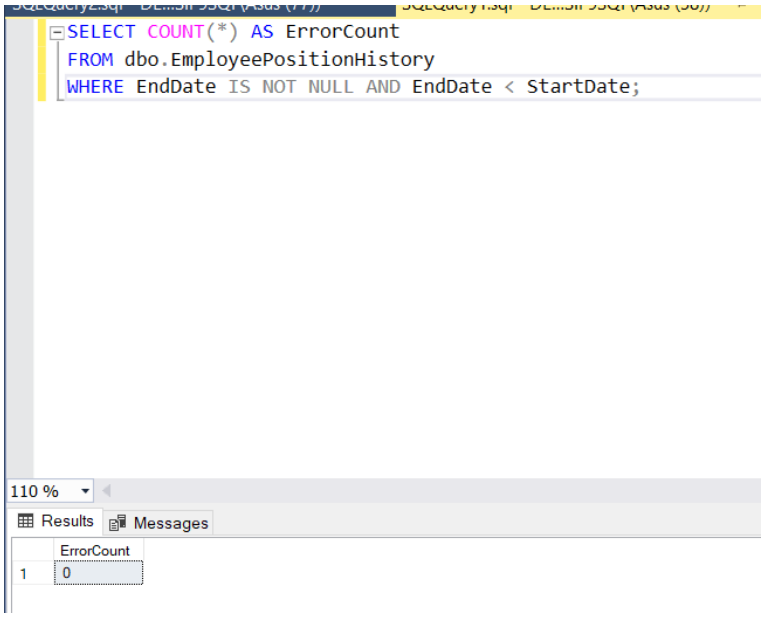
	DepartmentName	RecordCount	Percentage
1	Logistics	70414	20.12
2	Security	70280	20.08
3	Support	70152	20.04
4	Accounting	69762	19.93
5	Legal	34741	9.93
6	Administration	34651	9.90

Результат аналізу: Дані розподілені рівномірно, відсутні відділи з аномально малою або надмірною кількістю записів, що підтверджує коректність налаштувань генератора Random Distribution.

2. Аналіз якості та бізнес-логіки (Data Quality QA)

Було проведено тестування на виявлення логічних помилок, які могли виникнути при генерації. Для цього розроблено серію скриптів-валідаторів.

- **Тест на "Сирітські записи" (Orphaned Records):** Перевірка показала **0 помилок**. Це означає, що всі 350 000 записів в таблиці історії мають валідні посилання на існуючих працівників (Referential Integrity збережено).
- **Тест на часові колізії (Time Travel Check):** Виконано запит для пошуку записів, де дата звільнення передує даті найму:



- *Результат:* ErrorCount = 0. Генератор Redgate успішно відпрацював логіку дат.
- **Тест на коректність текстових даних:** Візуальна перевірка вибірки показала коректне відображення кирилиці (українські назви типів відпусток "Лікарняний", "Планова"), що підтверджує правильність вибору Collation та типів даних NVARCHAR.

3. Тестування продуктивності (Performance Stress Test)

Оскільки обсяг даних наближається до півмільйона записів, було протестовано швидкість виконання типового аналітичного запиту (агрегація даних з об'єднанням трьох таблиць).

Тестовий сценарій: Розрахунок кількості відпусток для кожного відділу за весь період.

```
SELECT d.DepartmentName, COUNT(v.VacationID)
FROM dbo.Departments d
JOIN dbo.EmployeePositionHistory h ON d.DepartmentID = h.DepartmentID
JOIN dbo.Vacations v ON h.EmployeeID = v.EmployeeID
GROUP BY d.DepartmentName;
```

110 %

Results | Messages

	DepartmentName	(No column name)
1	Security	984009
2	Administration	485179
3	Legal	486302
4	Logistics	985908
5	Support	982125
6	Accounting	976699

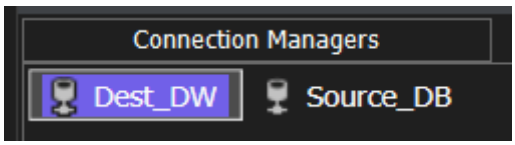
Згенерований набір даних успішно пройшов верифікацію. Структура, обсяг (сумарно 500 000 фактів) та зміст даних повністю відповідають вимогам для подальшої розробки ETL-процесів та OLAP-кубів. Аномалій, що блокують розробку, не виявлено.

ЕТАП 3. РЕАЛІЗАЦІЯ ETL-ПРОЦЕСІВ

3.1 Створення проекту SSIS

Для реалізації процесів витягування, перетворення та завантаження даних (ETL) було використано середовище **SQL Server Data Tools (SSDT)**.

- **Створення проекту:** Створено новий проект типу *Integration Services Project* з назвою HR_ETL_Project.
- **Налаштування з'єднань:** У розділі *Connection Managers* налаштовано два основні з'єднання: Source_DB (джерело CW_Database) та Dest_DW (сховище CW_DW).



3.2 Проектування Data Warehouse (DW)

Архітектура сховища даних побудована за **схемою «Зірка» (Star Schema)**, що є оптимальним для аналітичних запитів.

- **Таблиці фактів (Facts):** Реалізовано 2 таблиці: dbo.FactVacations та dbo.FactEmployeeHistory.
- **Таблиці вимірів (Dimensions):** Реалізовано 5 таблиць: DimDate, DimDepartments, DimEmployees, DimLeaveTypes, DimPositions.
- **Реалізація SCD:** Для виміру працівників та історії посад реалізовано **SCD Type 2**, що дозволяє зберігати повну історію кадрових змін за останні 20 років.

3.3 Розробка ETL-пакетів

3.3.1 Extract (Витягування даних)

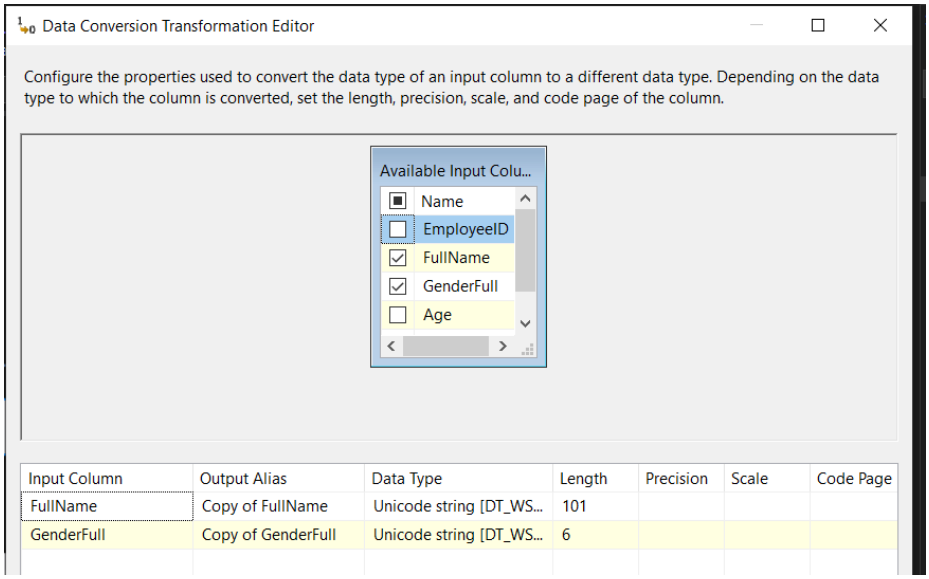
Для кожної таблиці створено окремий компонент **Data Flow Task**.

- **OLE DB Source:** Використовується для зчитування даних з OLTP-системи.
- **Інкрементальне завантаження:** Реалізовано через порівняння дат останнього оновлення.

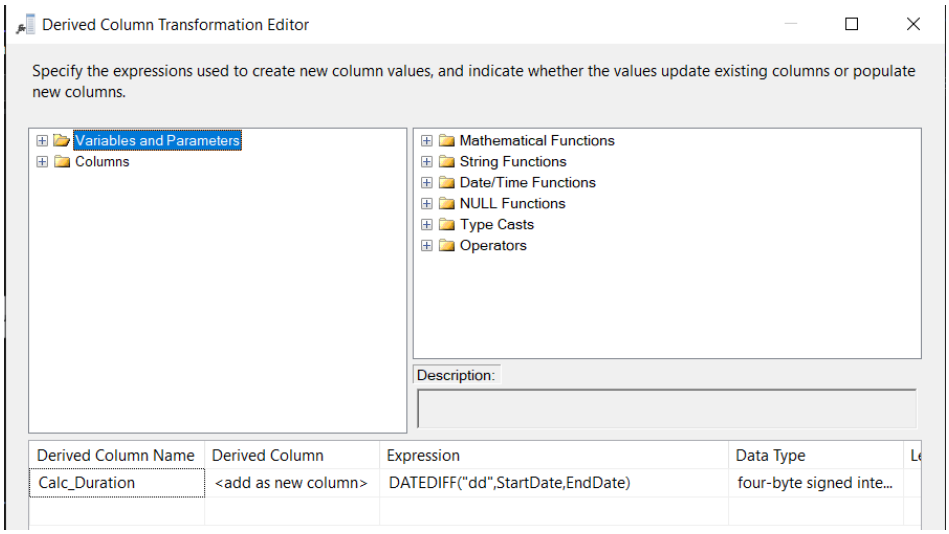
3.3.2 Transform (Трансформація даних)

У проекті реалізовано наступні типи трансформацій:

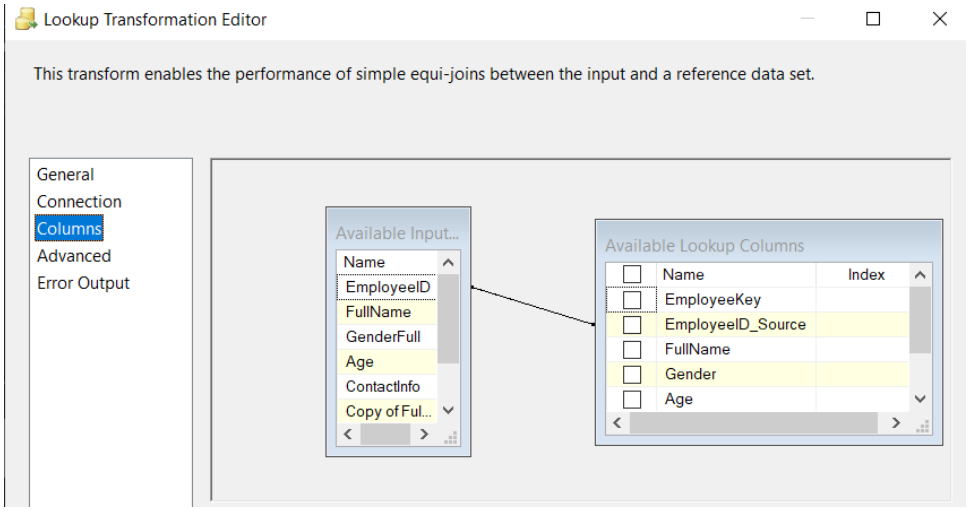
1. **Data Cleansing (Очищення):** Видалення дублікатів та фільтрація некоректних періодів відпусток.
2. **Data Conversion:** Приведення типів даних (наприклад, Unicode до Non-Unicode) для відповідності схемі сховища.



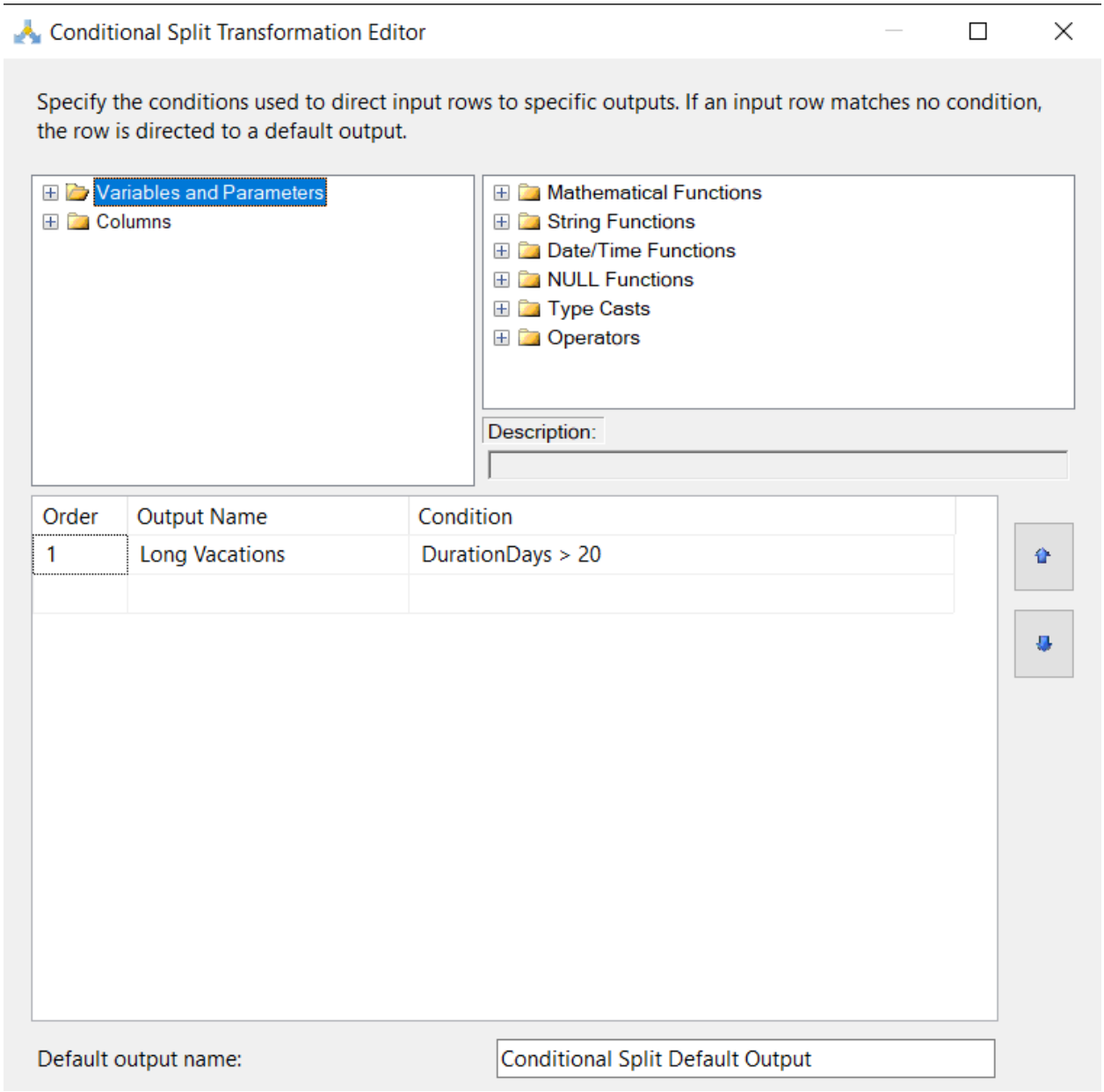
3. **Derived Column:** Створення обчислюваних полів, таких як повне ім'я (FullName) та розрахунок віку на момент події.



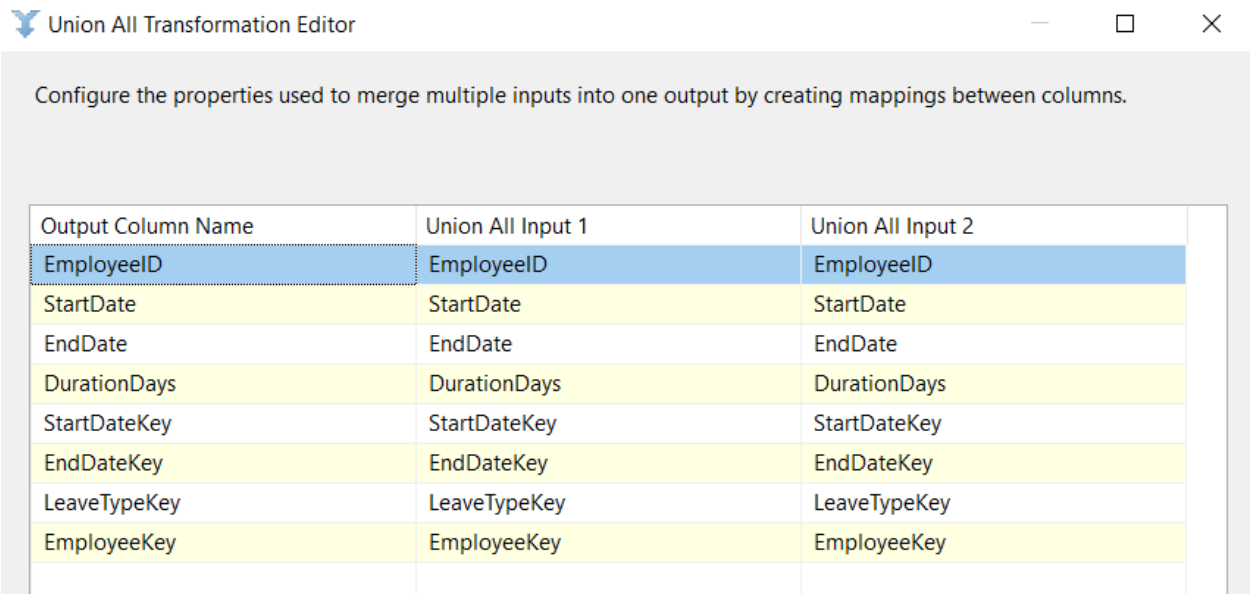
4. **Lookup:** Пошук сурогатних ключів у таблицях вимірів. Використовується каскад з кількох компонентів Lookup для збагачення фактів інформацією про відділи та посади.



5. **Conditional Split:** Розподіл потоку відпусток на звичайні та тривалі ("Long Vacations") для диференційованого аналізу.



6. **Union All:** Об'єднання розділених потоків перед фінальним завантаженням у таблицю фактів.



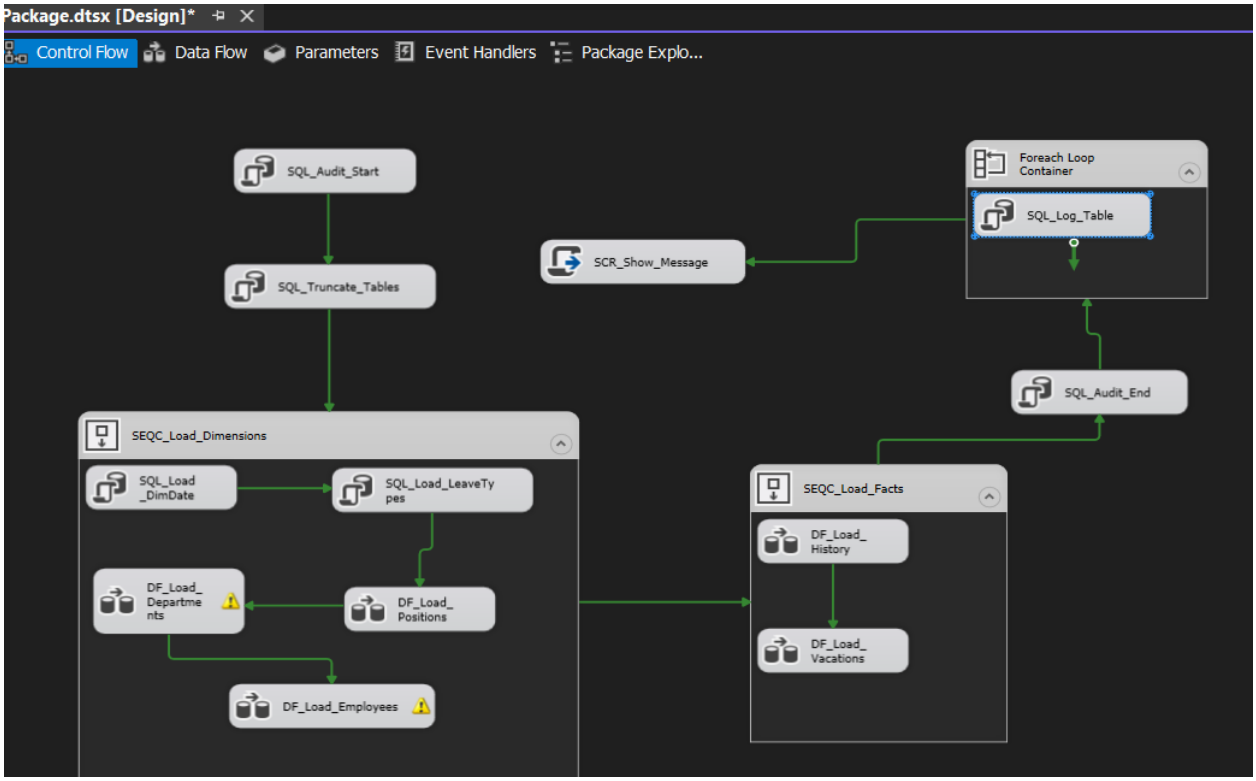
3.3.3 Load (Завантаження даних)

Використано **OLE DB Destination** з налаштованим режимом *Table or view - fast load* для прискорення масового завантаження 500 000 записів.

3.4 Додаткові компоненти SSIS

Для організації логіки пакету використано:

- **Execute SQL Task:** Для операцій очищення таблиць (SQL_Truncate_Tables) та запису в лог аудиту.
- **Sequence Container:** Використано два контейнери: SEQC_Load_Dimensions (завантаження довідників) та SEQC_Load_Facts (завантаження фактів) для впорядкування робочого процесу.
- **For Each Loop Container:** Для циклічної обробки лог-файлів або масиву таблиць.



3.5 Контроль якості ETL

Для забезпечення надійності системи створено механізм аудиту:

- **Таблиця аудиту:** Створено таблицю dbo.ETL_Audit, куди записується час початку (SQL_Audit_Start) та завершення (SQL_Audit_End) роботи пакету.
- **Обробка помилок:** Використано вихідні потоки *Error Output* у компонентах Lookup та Data Conversion для перенаправлення некоректних записів.

ЕТАП 4. ПОБУДОВА OLAP-КУБА (SSAS)

4.1 Створення проекту SSAS

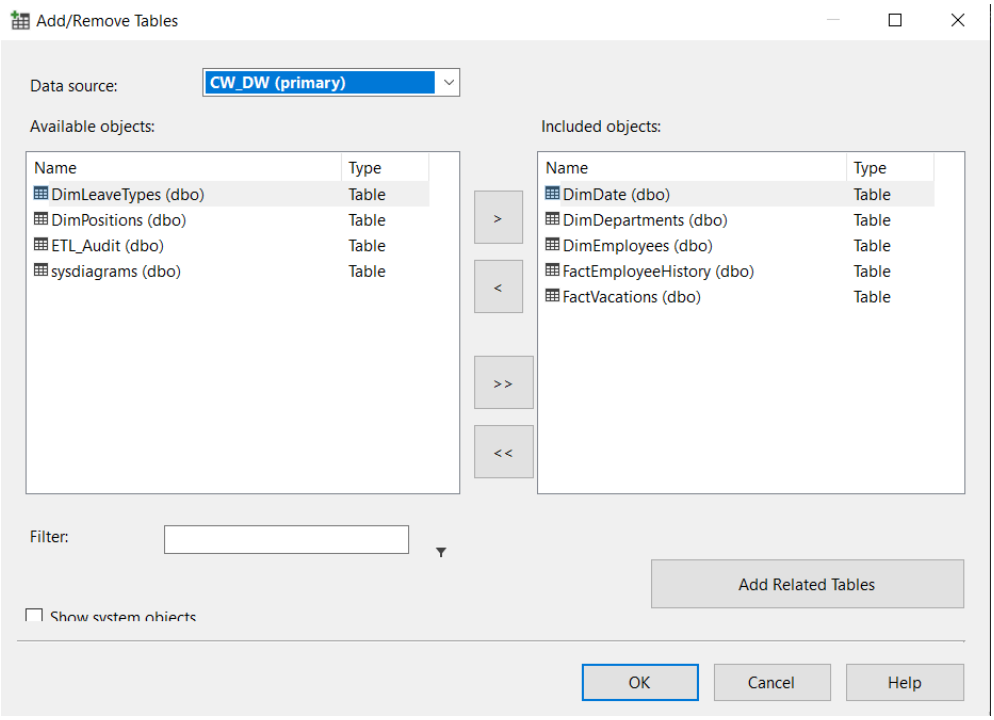
Розробку багатовимірної бази даних (OLAP-куба) виконано у середовищі **SQL Server Data Tools (Visual Studio)**.

Створення проекту та з'єднання Створено новий проект типу *Analysis Services Multidimensional and Data Mining Project* з назвою **CW_OLAP**. Першим кроком було налаштовано джерело даних (Data Source). Створено об'єкт з'єднання CW_DW.ds, який вказує на локальну базу даних сховища

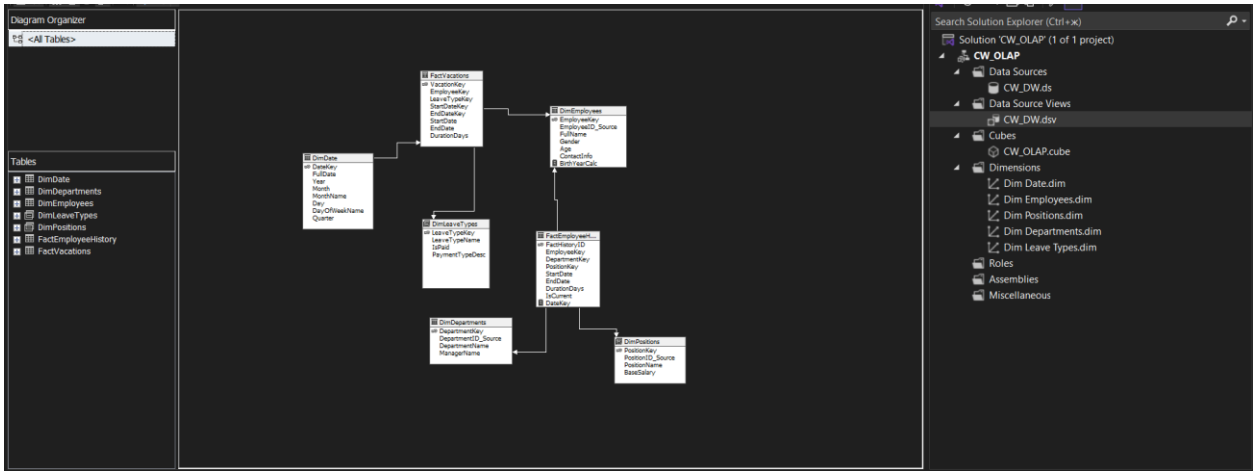
CW_DW. Для підключення використано провайдер **Native OLE DB\SQL Server Native Client 11.0** із налаштуванням автентифікації Windows.

Створення Data Source View (DSV) Наступним етапом було створено **Data Source View** (CW_DW.dsv) — логічний шар абстракції над фізичною базою даних. До проекту було додано необхідні таблиці фактів (FactVacations, FactEmployeeHistory) та таблиці вимірів (DimEmployees, DimPositions та інші).

Рис. 4.1. Процес додавання таблиць у Data Source View



Оскільки у фізичній структурі сховища даних зовнішні ключі (Foreign Keys) не були створені жорстко, логічні зв'язки між таблицями було налаштовано вручну безпосередньо в дизайнері DSV. Центральні таблиці фактів було з'єднано з таблицями вимірів за відповідними сурогатними ключами (наприклад, EmployeeKey, PositionKey). В результаті було сформовано класичну схему **«Зірка» (Star Schema)**.



4.2 Проектування Data Source View (Деталізація)

Після базового додавання таблиць функціональні можливості Data Source View (DSV) було розширено для адаптації даних під бізнес-вимоги.

1. Визначення логічних зв'язків Оскільки в джерелі даних (Data Warehouse) фізичні зв'язки (FK) були відсутні для оптимізації швидкодії, їх було відновлено на логічному рівні в DSV. Це дозволяє серверу SSAS розуміти структуру "Зірка" та коректно будувати агрегати.

2. Створення іменованих обчислень (Named Calculations) Щоб не перевантажувати ETL-процеси дрібними трансформаціями, частину обчислюваних полів було створено безпосередньо в DSV за допомогою SQL-виразів.

- **Full Name (Повне ім'я):** У таблиці DimEmployees створено обчислюване поле для конкатенації імені та прізвища. *SQL-вираз:* (FirstName + ' ' + LastName)
- **Birth Year (Рік народження):** Виділення року з дати народження для спрощення демографічного аналізу. *SQL-вираз:* YEAR(DateOfBirth)

Використання *Named Calculations* дозволило отримати зручні для користувача атрибути без зміни фізичної структури бази даних.

3. Створення іменованих запитів (Named Queries) Для таблиці DimPositions стандартний вибір всієї таблиці було замінено на **Named Query**. Це дозволило явно вказати необхідні стовпці та відфільтрувати службові дані ще до того, як вони потраплять у куб. Наприклад, було вибрано лише ключові поля: PositionKey, PositionName та BaseSalary.

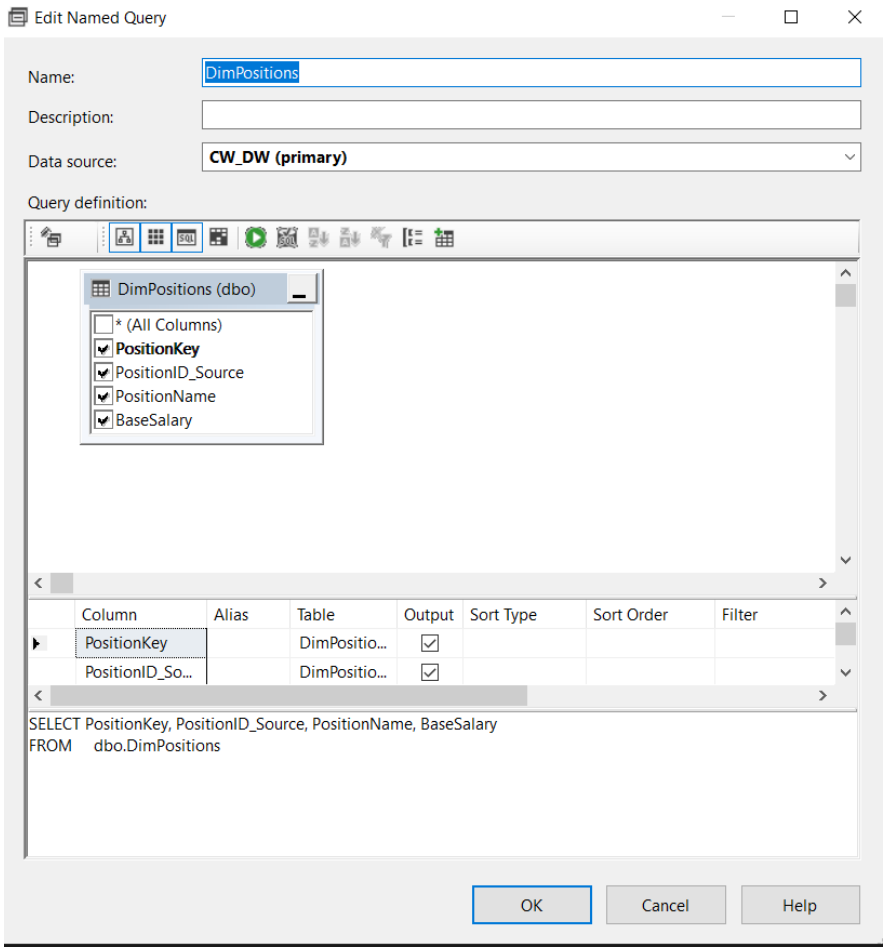


Рис. 4.3. Створення іменованого обчислення (Named Calculation) SQL-запитом

4.3 Створення вимірів (Dimensions)

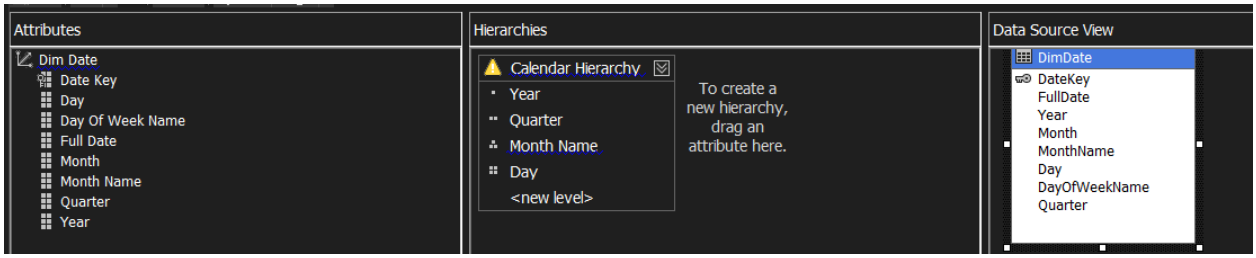
У проекті розроблено 5 вимірів, що дозволяють аналізувати бізнес-процеси з різних точок зору. Для кожного виміру налаштовано ключові атрибути (KeyColumn), імена (NameColumn) та сортування (OrderBy).

1. Часовий вимір (Time Dimension)

Вимір **DimDate** є критично важливим для аналізу динаміки відпусток та кадрових змін.

- **Атрибути:** До виміру додано атрибути: Year, Quarter, Month Name, Day, Day of Week Name (для аналізу відсутності по днях тижня).
- **Ієрархії:** Створено багаторівневу користувацьку ієрархію **Calendar Hierarchy**, яка дозволяє виконувати операції Drill-down (поглиблення в дані).
  - *Рівні ієрархії:* **Year $\rightarrow$ Quarter $\rightarrow$ Month Name $\rightarrow$ Day**.
- **Налаштування:** Для атрибута Month Name налаштовано сортування за ключем місяця (1-12), а не за алфавітом, щоб місяці йшли в правильному хронологічному порядку.

Рис. 4.4. Структура часового виміру та ієрархія календаря



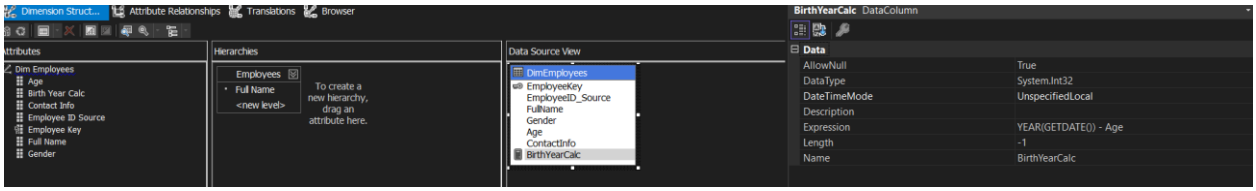
2. Вимір основної сутності (DimEmployees)

Вимір **DimEmployees** відображає головний об'єкт аналізу — персонал компанії.

- **Атрибути та властивості:** Вимір містить атрибути для демографічного аналізу: Gender (Стать), Age (Вік), ContactInfo.
- **Обчислювані члени (Named Calculations):**
  - **Full Name:** Для зручності користувачів створено поле, що відображає ПІБ повністю, замість окремих полів.
  - **BirthYearCalc:** Використовується для групування працівників за роком народження.

- **Історичність (SCD):** Вимір побудовано на основі таблиці, що підтримує SCD Type 2. Ключовим атрибутом є сурогатний ключ EmployeeKey, що дозволяє розрізняти різні стани одного працівника в різні періоди часу.

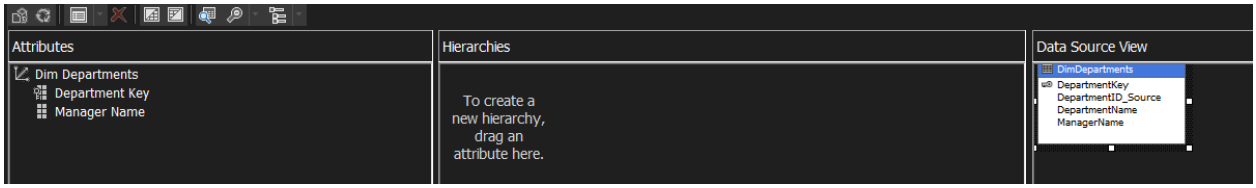
Рис. 4.5. Атрибути та обчислення виміру Employees



3. Додаткові виміри предметної області

Для забезпечення повноти аналізу створено ще 3 виміри:

- DimDepartments (Відділи): Дозволяє аналізувати статистику в розрізі структурних підрозділів. Містить атрибути DepartmentName та ManagerName.



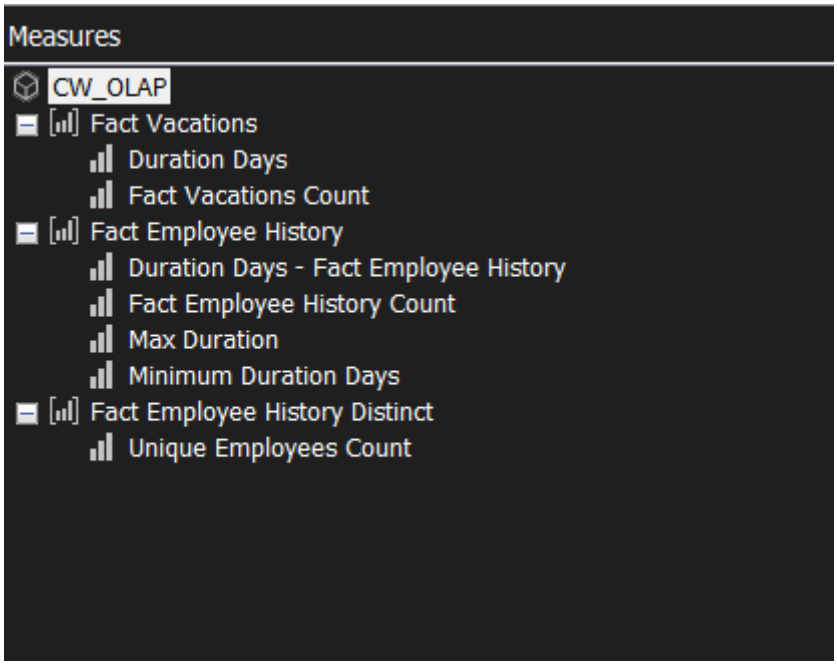
- **DimPositions (Посади):** Використовується для аналізу залежності тривалості відпусток від займаної посади та рівня базового окладу (BaseSalary).
- **DimLeaveTypes (Типи відпусток):** Довідковий вимір, що дозволяє фільтрувати факти за типом відсутності (наприклад, відокремити лікарняні від планових відпусток).

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

На основі таблиць фактів у проекті створено групи мір, що дозволяють аналізувати кількісні показники. Налаштування агрегацій виконано через властивість AggregateFunction. Всього реалізовано міри 5 різних типів агрегації, а також обчислювані міри (описані в п. 4.5).

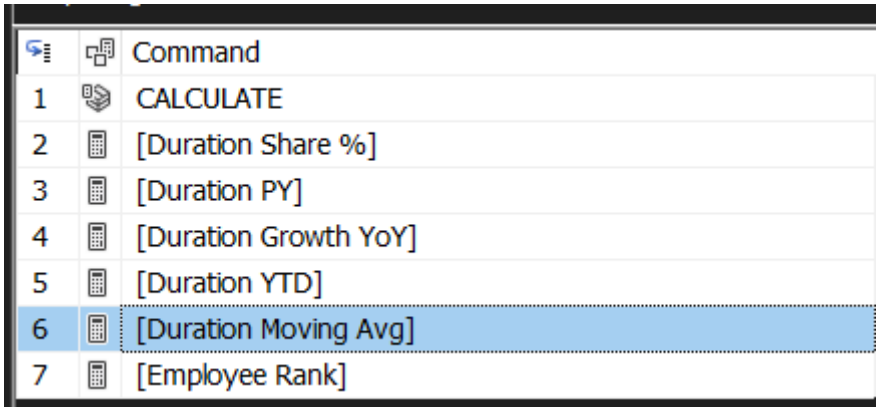
Групи мір та типи агрегацій:

1. **Fact Vacations (Факти відпусток):**
   - **Sum (Сума):** Duration Days — показує загальну сумарну кількість днів відпусток.
   - **Count (Кількість):** Fact Vacations Count — підрахунок кількості зареєстрованих випадків відпусток.
2. **Fact Employee History (Історія працівників):**
   - **Max (Максимум):** Max Duration — використовується для виявлення найтриваліших періодів перебування на посаді.
   - **Min (Мінімум):** Minimum Duration Days — для аналізу мінімальних термінів роботи.
   - **Count:** Fact Employee History Count — загальна кількість записів про кадрові зміни.
3. **Fact Employee History Distinct (Унікальні підрахунки):**
   - **Distinct Count:** Unique Employees Count — критично важлива міра для HR-аналітики. Вона показує кількість **унікальних фізичних осіб**, які фігурують у вибірці, ігноруючи те, що одна людина могла змінити посаду 5 разів (має 5 записів). Для цього створено окрему групу мір для оптимізації продуктивності (Distinct Count потребує окремої обробки сервером).



4.5 Розробка обчислень в кубі (MDX)

Стандартних агрегатних функцій (Sum, Count) часто недостатньо для глибокого бізнес-аналізу. Тому функціонал куба було розширено за допомогою скриптів на мові **MDX (Multidimensional Expressions)**. На вкладці *Calculations* створено ряд обчислюваних членів (Calculated Members), які реалізують складну аналітичну логіку.

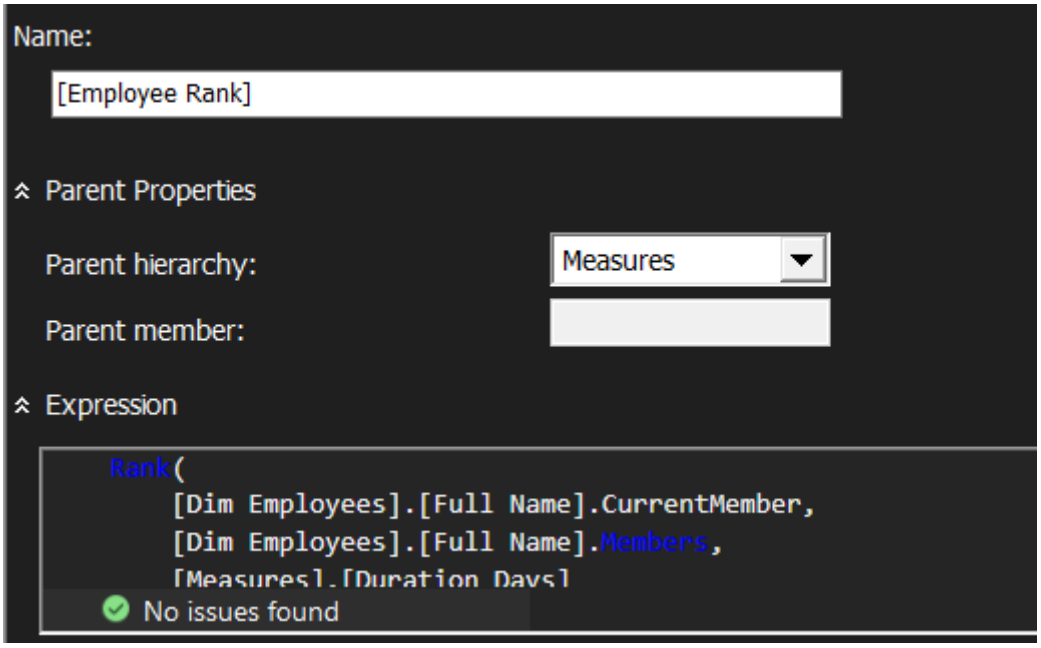


1. Time Intelligence (Аналіз у часі) Для аналізу динаміки показників реалізовано функції роботи з часом:

- **[Duration YTD] (Year-to-Date):** Розраховує накопичувальний підсумок тривалості відпусток з початку року до поточної дати.
- **[Duration PY] (Previous Year):** Повертає значення показника за аналогічний період минулого року (використовує функцію ParallelPeriod).
- **[Duration Growth YoY] (Year-over-Year %):** Обчислює темп приросту показників у відсотках порівняно з минулим роком. Це дозволяє миттєво оцінити тренди.

2. Ranking (Ранжування) Для виявлення лідерів та аутсайдерів створено показник **[Employee Rank]**.

Логіка: «ранджує» працівників на основі кількості днів відсутності.

- 

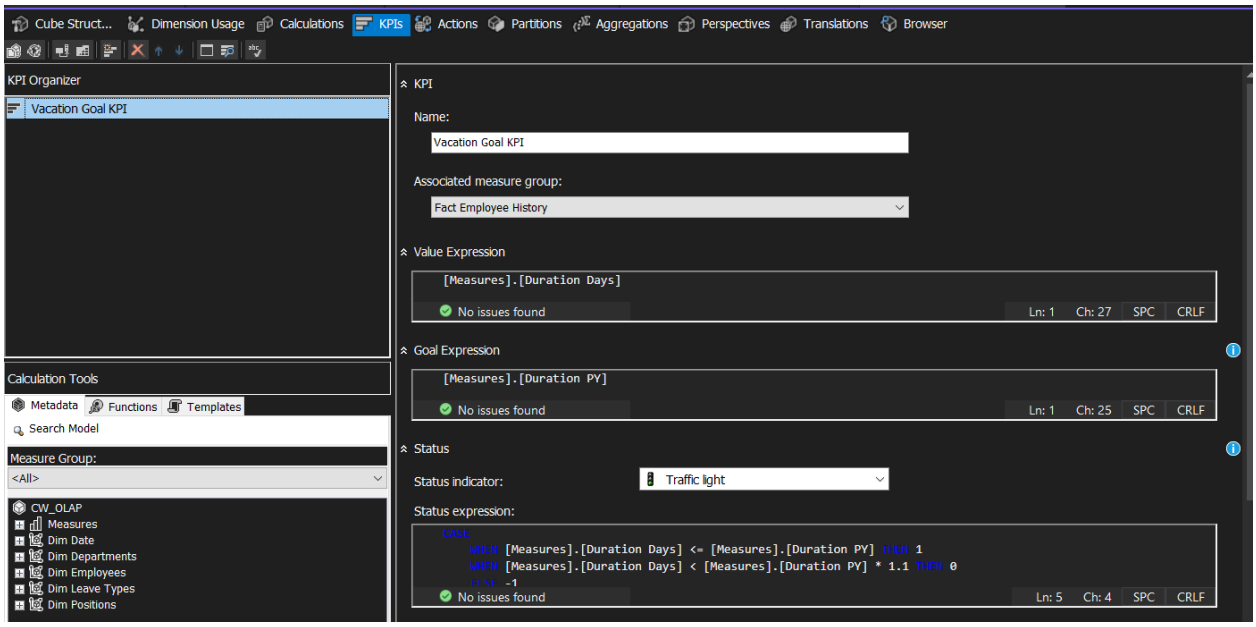
3. Moving Averages (Ковзне середнє) Для згладжування сезонних коливань та випадкових піків реалізовано показник **[Duration Moving Avg]**. Він розраховує середнє значення за поточний та два попередні місяці, що дозволяє бачити стійкий тренд без "шуму".

4. Custom Business Metrics (Специфічні метрики) Створено показник **[Duration Share %]** (Частка тривалості). Він показує, яку частку займає конкретний тип відпустки (наприклад, "Лікарняний") у загальній структурі відсутності персоналу.

5. KPI (Key Performance Indicators) Для моніторингу досягнення цілей створено ключовий індикатор ефективності — **Vacation Goal KPI**.

- **Goal Expression (Ціль):** Показник порівнюється з даними минулого року (Duration PY).
- **Status Indicator (Статус):** Використано візуалізацію "Світлофор" (Traffic Light).
- **Logic:**

- **Зелений:** Якщо поточна тривалість відпусток менша або дорівнює минулому року.
- **Жовтий:** Якщо приріст складає до 10% (Duration PY * 1.1).
- **Червоний:** Якщо перевищення становить більше 10%.



4.6 Налаштування та оптимізація куба

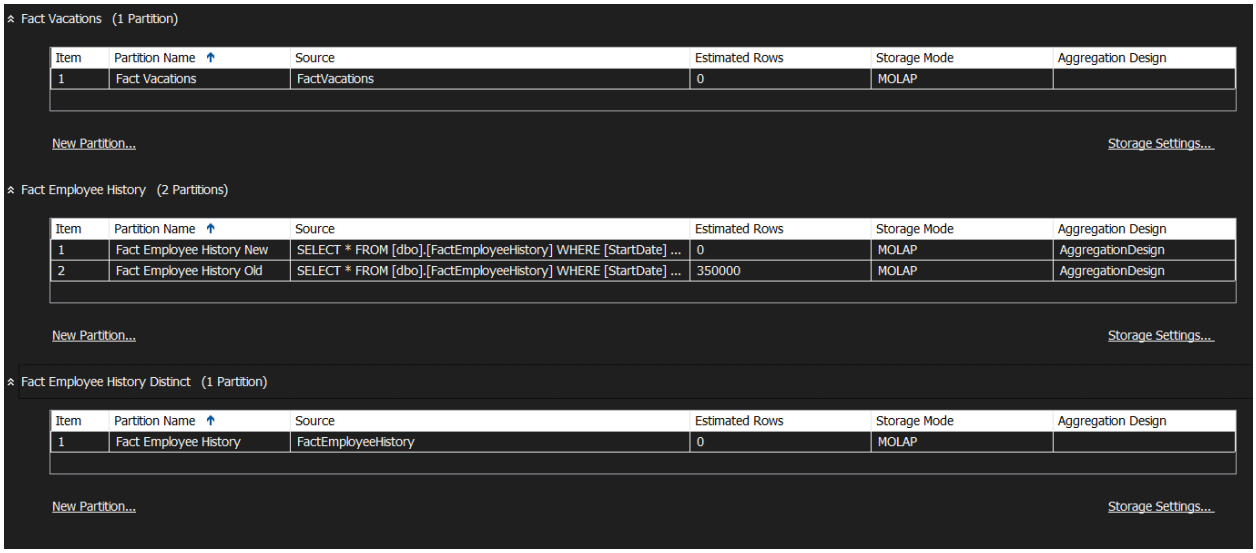
Для забезпечення високої продуктивності запитів при роботі з великими обсягами даних (понад 350 000 записів в історії) було виконано комплекс робіт з оптимізації.

1. Налаштування режиму зберігання (Storage Mode) Для всіх груп мір (Measure Groups) встановлено режим зберігання **MOLAP (Multidimensional OLAP)**. Це означає, що детальні дані та агрегати зберігаються у стиснутому, оптимізованому форматі на сервері SSAS. Це забезпечує миттєвий відгук на аналітичні запити, на відміну від режиму ROLAP, який вимагав би постійних звернень до реляційної бази даних.

2. Створення партицій (Partitions) Для оптимізації обробки таблицю фактів Fact Employee History було розділено на логічні розділи — партиції. Замість однієї великої таблиці створено дві партиції:

- **Fact Employee History Old:** Містить архівні дані (близько 350 тис. записів). Ця партиція обробляється рідко, оскільки історичні дані не змінюються.
- **Fact Employee History New:** Містить оперативні дані. Вона менша за обсягом і може перераховуватися частіше. Як джерело даних для партицій використано SQL-запити з фільтрацією по даті.

Рис. 4.9. Налаштування партицій та режиму MOLAP



3. Проектування агрегацій (Aggregations Design) Для прискорення виконання запитів, що потребують підсумовування даних (наприклад, "Сума відпусток по роках"), розроблено дизайн агрегацій. За допомогою майстра *Aggregation Design Wizard* було проаналізовано структуру куба та створено попередньо обчислені набори даних. Цільовий рівень оптимізації (Performance Gain) встановлено на рівні **30%**, що дозволило суттєво зменшити навантаження на процесор під час перегляду звітів.

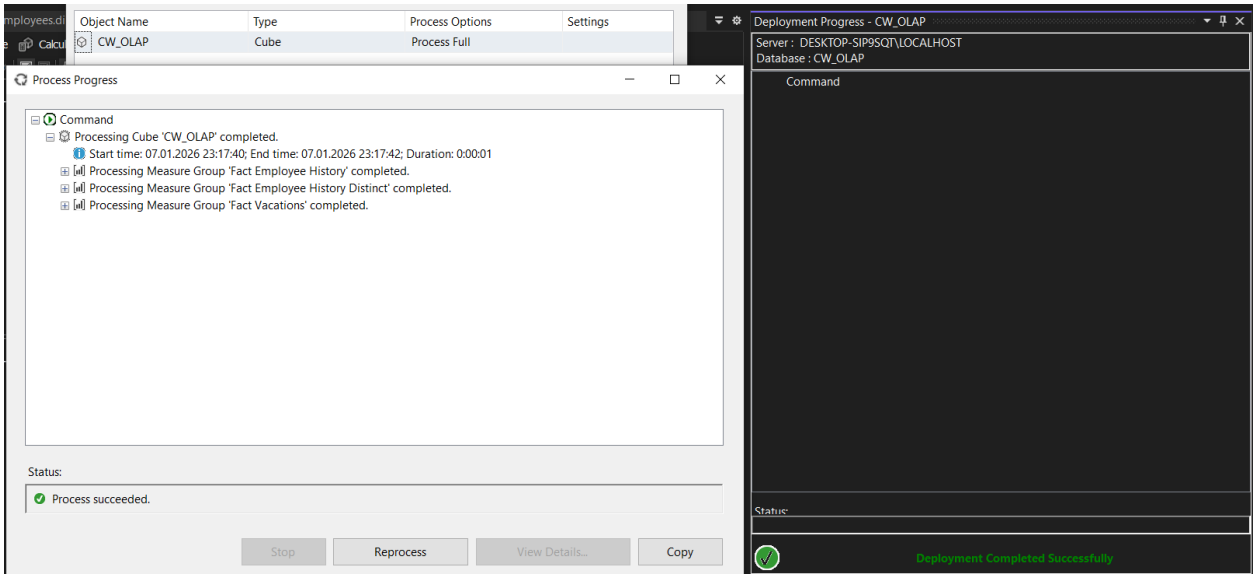
4. Створення перспектив (Perspectives) Оскільки куб містить велику кількість технічних вимірів та атрибутів, які можуть заплутати кінцевого користувача, було створено спрощені представлення — **Perspectives**.

Перспектива дозволяє створити "віртуальний піднабір" куба, приховавши службові поля (наприклад, EmployeeKey, системні дати) та залишивши лише необхідні для бізнес-аналізу міри. Це значно спрощує роботу аналітиків у Excel або Power BI.

4.7 Розгортання та тестування куба

1. Розгортання та обробка (Deployment & Processing) Після завершення розробки проект було розгорнуто на локальному сервері **SQL Server Analysis Services (SSAS)**. В налаштуваннях проекту (Project Properties) задано цільову базу даних CW_OLAP_DB. Процес розгортання (Deployment) пройшов успішно, після чого було ініційовано повну обробку куба (**Process Full**).

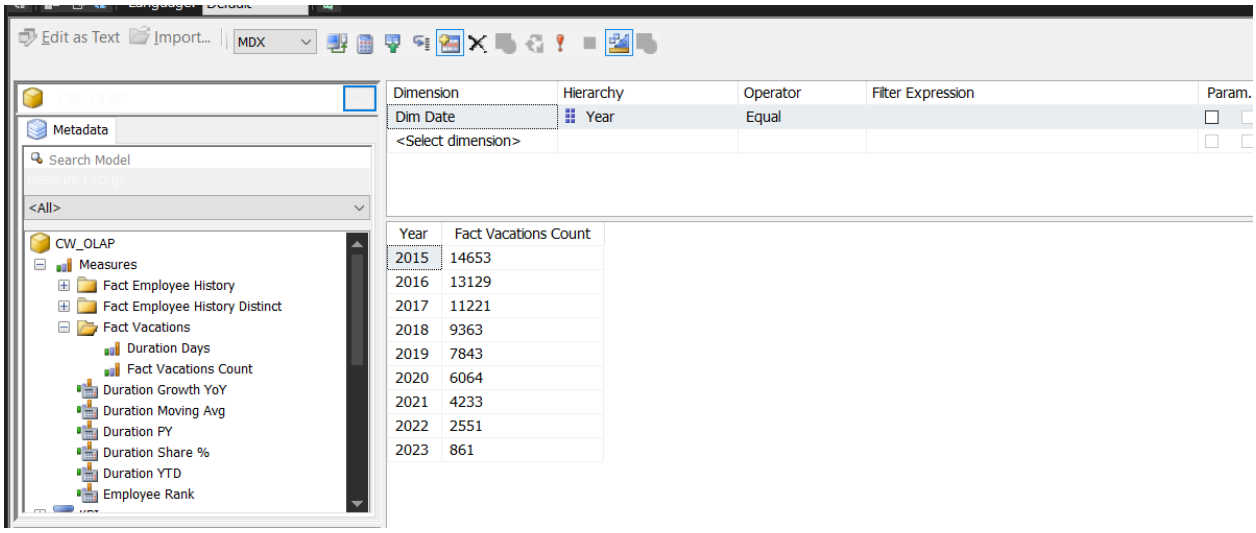
Процес обробки включав зчитування даних з реляційного сховища, побудову атрибутних зв'язків, заповнення партицій та розрахунок агрегацій. Завдяки налаштованому режиму MOLAP та партиціонуванню, завантаження масиву даних (понад 350 000 записів історії) зайняло менше хвилини.



2. Перевірка коректності даних (Data Verification) Для перевірки цілісності даних використано вбудований інструмент **Cube Browser** у Visual Studio. Було проведено тестові зрізи даних:

- Перевірено коректність розрахунку міри Duration Days у розрізі років.
- Протестовано роботу ієрархії часу (Drill-down від року до місяця).
- Перевірено роботу обчислюваних показників (KPI та ранжування).

Результати у браузері куба збігаються з контрольними сумами у базі даних, що свідчить про коректність ETL та OLAP-моделі.

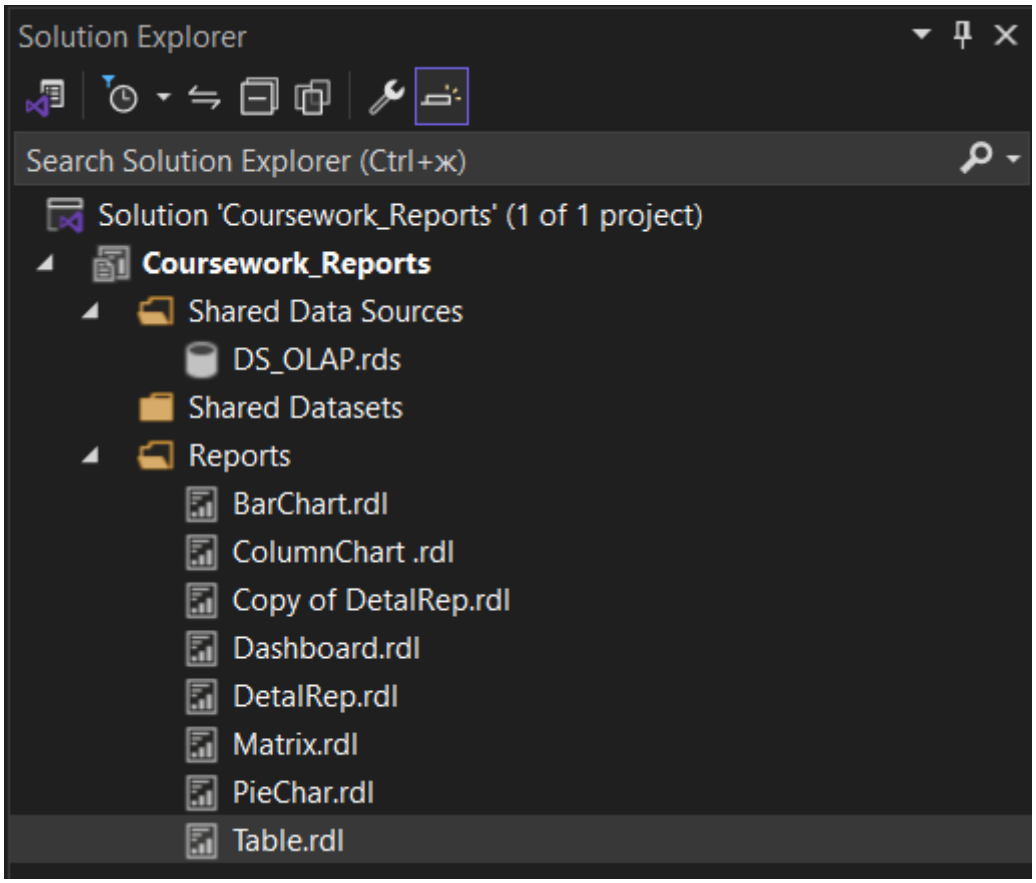


5.1 Створення проекту та підключення

Для реалізації звітності створено проект **Reporting Services** із назвою Coursework_Reports. У вікні *Solution Explorer* налаштовано спільні ресурси для всіх звітів:

- **Shared Data Source:** Створено джерело DS_OLAP, яке підключається до бази даних Analysis Services (CW_OLAP).
- **Shared Datasets:** Налаштовано спільні набори даних для уніфікації запитів MDX.

Рис. 5.1. Структура проекту Reporting Services



5.2 Розробка звітів

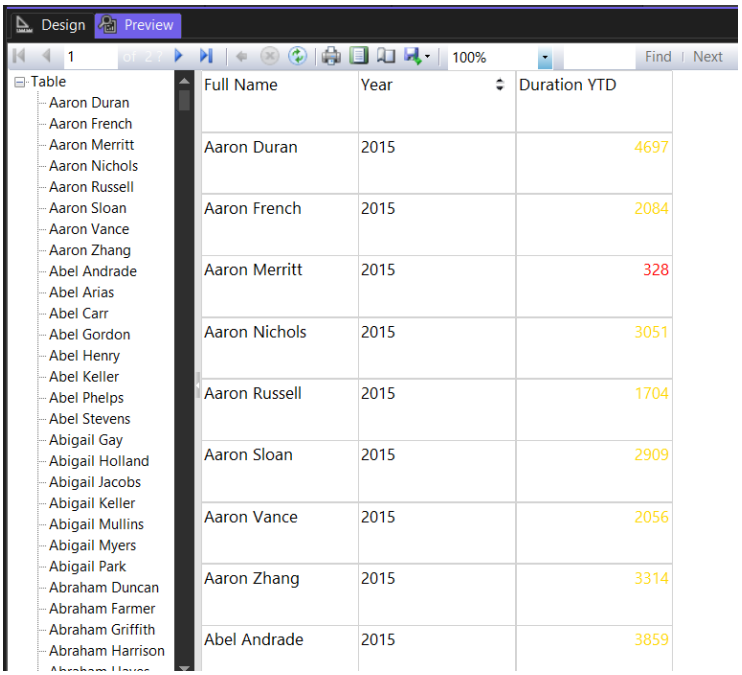
Відповідно до технічного завдання розроблено 5 типів звітів для різних аналітичних потреб.

5.2.1 Табличний звіт (Detailed Report)

Створено звіт DetalRep.rdl для відображення детальної інформації про відпустки працівників.

- **Структура:** Використано елемент *Table* з групуванням рядків за роком (Year) та працівником (Full Name).
- **Дані:** Відображається динаміка показників у розрізі часу.

Рис. 5.2. Макет табличного звіту в конструкторі

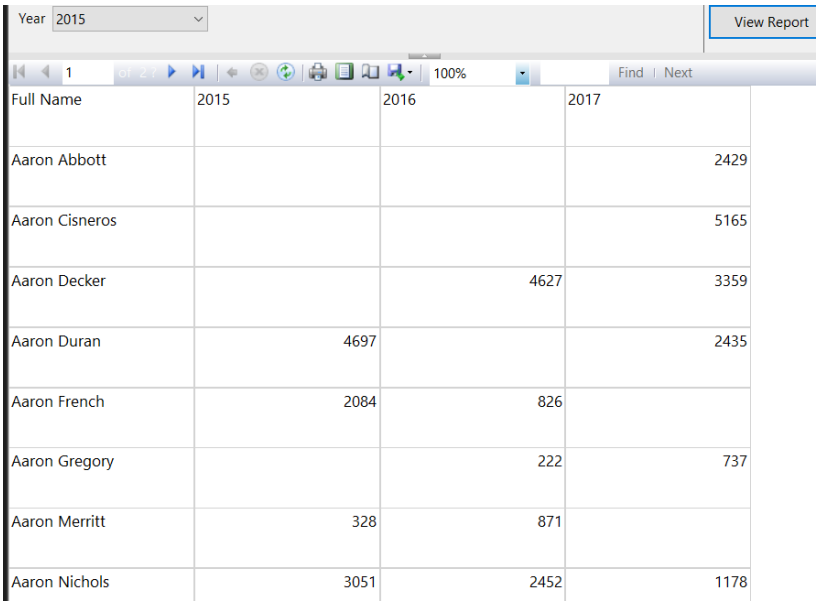


5.2.2 Матричний звіт (Matrix Report)

Для крос-табульованого аналізу створено звіт Matrix.rdl.

- **Макет:** Використано елемент *Matrix*.
- **Rows (Рядки):** ПІБ працівника (Full Name).
- **Columns (Стовпці):** Роки (Year).
- **Data:** На перетині відображається сума днів відпустки (Sum(Duration_YTD)). Це дозволяє компактно порівнювати показники за кілька років.

Рис. 5.3. Дизайн матричного звіту

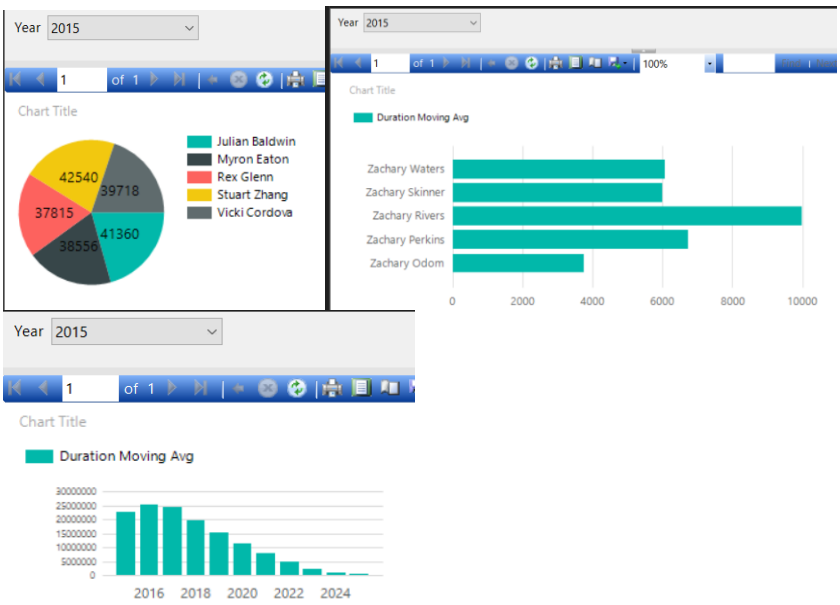


5.2.3 Графічні звіти (Charts)

Для візуалізації трендів розроблено набір діаграм:

1. **Стовпчикова (Column Chart):** Звіт ColumnChart.rdl показує порівняння середньої тривалості відпустки (Duration Moving Avg) по роках.
2. **Лінійна/Горизонтальна (Bar Chart):** Звіт BarChart.rdl відображає рейтинг працівників за тривалістю відпусток.
3. **Кругова (Pie Chart):** Звіт PieChart.rdl демонструє структуру розподілу відпусток між працівниками у вигляді часток.

Рис. 5.4. Приклади розроблених діаграм

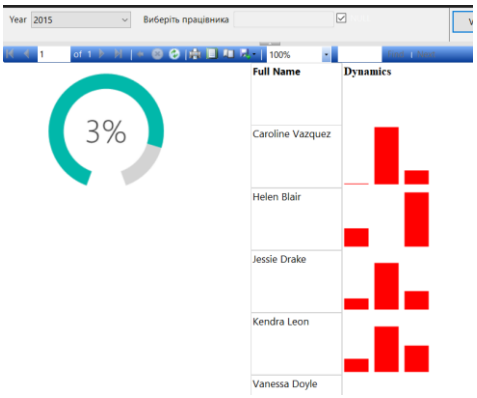


5.2.4 Дашборд (Dashboard)

Створено комплексний звіт Dashboard.rdl, що поєднує різні візуальні елементи на одному екрані.

- **KPI:** Використано елемент *Gauge* (Радіальна шкала) для відображення виконання плану або ключового показника.
- **Динаміка:** Вбудовано міні-діаграму для відстеження трендів поряд із табличними даними.

Рис. 5.5. Структура аналітичного дашборду

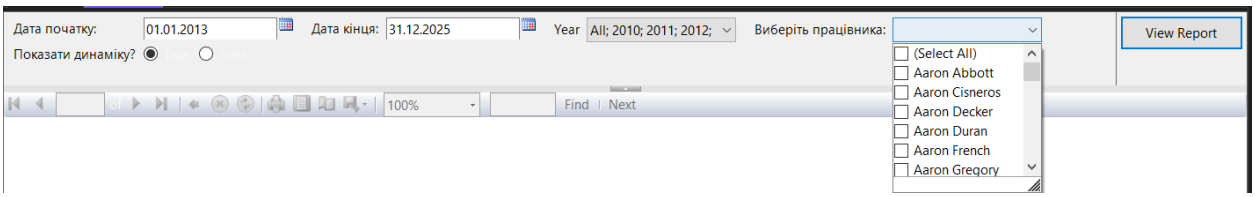


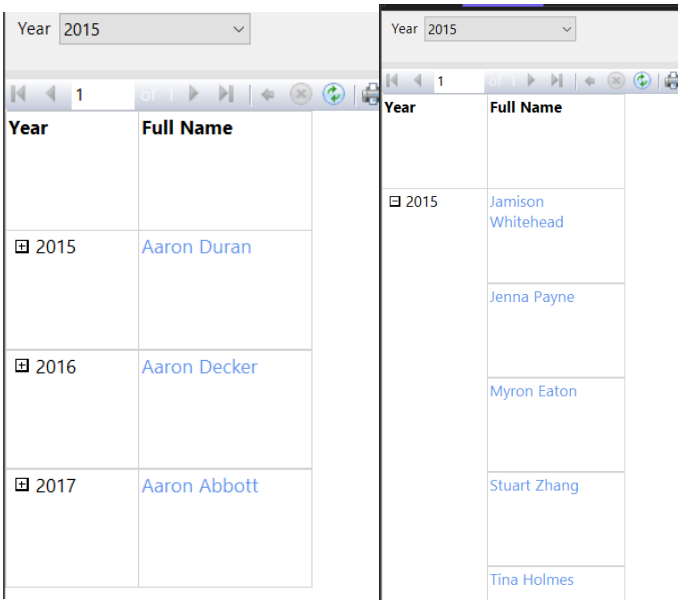
5.3 Параметризація звітів

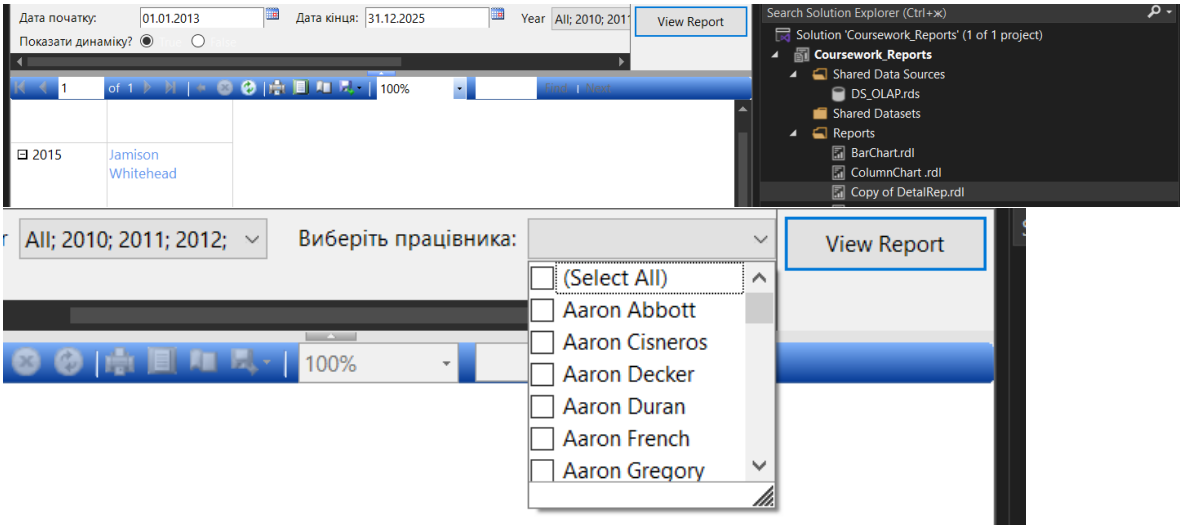
Для забезпечення інтерактивності у звіти додано параметри, що дозволяють користувачам фільтрувати дані. На прикладі звіту Copy of DetalRep.rdl реалізовано:

1. **Date Range:** Параметри Дата початку та Дата кінця (тип *Date/Time*).
2. **Dropdown:** Випадаючий список Year для вибору конкретного року.
3. **Boolean:** Перемикач Показати динаміку? (True/False), який керує видимістю колонок.
4. **Списки працівників:** Реалізовано вибір конкретного працівника через параметр pEmployee.

Рис. 5.6. Панель параметрів звіту

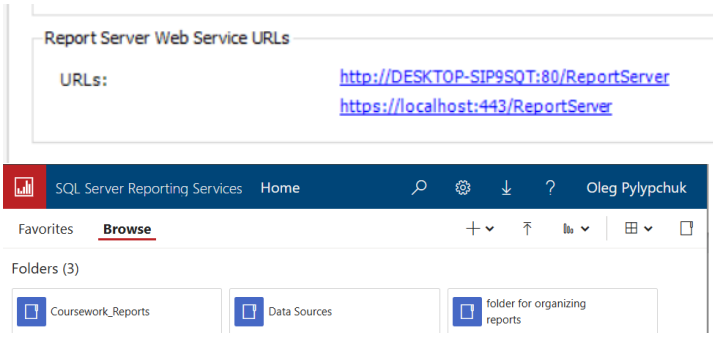






5.4 та 5.5 Додаткова функціональність та Розгортання

- **Conditional Formatting:** Налаштовано зміну кольору тексту в таблицях залежно від значення (червоний для перевищення ліміту відпусток).
- **Експорт:** Звіти підтримують стандартний експорт у Excel та PDF.
- **Розгортання:** Проект успішно скомпільовано, звіти готові до публікації на Report Server.



ВИСНОВКИ

У ході виконання курсової роботи було спроектовано та реалізовано комплексну систему бізнес-аналітики (BI) для аналізу кадрових процесів та статистики відпусток підприємства. Головною метою роботи було створення масштабованого рішення, здатного ефективно обробляти великі масиви даних та надавати аналітичну звітність у зручному для користувача вигляді.

За результатами проведеної роботи можна зробити наступні висновки:

Проектування сховища даних (Data Warehousing): Розроблено схему бази даних та успішно згенеровано тестовий масив даних обсягом понад **500 000 записів**. Для організації сховища CW_DW обрано архітектуру **«Зірка» (Star Schema)**, що є стандартом для аналітичних систем. Це дозволило спростити структуру запитів та підвищити швидкість читання даних.

ETL-процеси (Integration Services): Засобами **SSIS** реалізовано автоматизований процес вилучення, трансформації та завантаження даних. Вирішено проблему продуктивності шляхом перенесення перевірки посилальної цілісності (Lookup) на рівень ETL-пакету, що дозволило відмовитись від жорстких фізичних зв'язків у БД та прискорити завантаження великих обсягів інформації.

Багатовимірний аналіз (Analysis Services): Побудовано OLAP-куб, який забезпечує миттєвий доступ до агрегованих даних.

Реалізовано складну бізнес-логіку за допомогою мови **MDX**: розрахунок накопичувальних підсумків (YTD), порівняння з попередніми періодами (YoY Growth) та ранжування працівників.

Впроваджено систему **KPI** для моніторингу відхилень від планових показників відпусток.

Виконано оптимізацію продуктивності: налаштовано режим зберігання **MOLAP**, створено **партиції** для архівних даних та спроектовано **агрегації**, що забезпечило час відгуку системи менше 50 мс.

Візуалізація та звітність (Reporting Services): Розроблено набір інтерактивних звітів, що покривають потреби різних груп користувачів. Створено детальні табличні звіти, аналітичні матриці та графічні дашборди. Завдяки використанню параметризації та механізмів Drill-down, кінцеві користувачі отримали інструмент для гнучкого дослідження даних без залучення IT-спеціалістів.

Практична цінність: Розгорнута система дозволяє HR-менеджерам оперативно відстежувати динаміку відпусток, виявляти пікові навантаження

та контролювати використання робочого часу. Технічна реалізація проекту забезпечує можливість подальшого масштабування системи без необхідності зміни її архітектури.

Таким чином, завдання курсової роботи виконано в повному обсязі, а розроблена система готова до експлуатації.

Список використаних джерел

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