Міністерство освіти і науки України
Національний університет “Львівська політехніка”
Інститут прикладної математики та фундаментальних наук

Звіт
про виконання курсової роботи
з курсу “Надвеликі бази даних”

Виконала:
Кудрявцева О.М.
Перевірив:
Любінський Б.Б.

Львів 2025

Розроблено та спроєктовано інформаційно-аналітичну систему з використанням платформи Microsoft SQL Server. Система забезпечує централізоване зберігання даних, їх обробку та аналітичний аналіз із застосуванням сучасних технологій інтеграції, багатовимірного моделювання та візуалізації. Реалізоване рішення дозволяє отримувати узагальнену інформацію та проводити аналіз показників для підвищення ефективності прийняття управлінських рішень.

Was designed and implemented using the Microsoft SQL Server platform. The system provides centralized data storage, processing, and analytical analysis through the application of modern data integration, multidimensional modeling, and visualization technologies. The developed solution enables the generation of aggregated information and supports effective decision-making processes.

Зміст

Вступ

У сучасних умовах цифровізації суспільства інформація перетворилася на стратегічний ресурс, від ефективного використання якого залежить успішність діяльності організацій у різних галузях. Значні обсяги даних щоденно накопичуються в результаті операційної діяльності підприємств, установ та сервісів, що потребує впровадження надійних та ефективних інструментів для їх зберігання, обробки та аналізу. Саме тому розробка та впровадження інформаційно-аналітичних систем є актуальним завданням сучасної інформаційної інфраструктури.

Традиційні інформаційні системи, орієнтовані переважно на обробку транзакцій, не завжди здатні забезпечити глибокий аналіз накопичених даних, виявлення закономірностей і підтримку стратегічного управління. У зв’язку з цим виникла необхідність у використанні спеціалізованих аналітичних підходів, які дозволяють працювати з історичними даними, здійснювати багатовимірний аналіз та формувати узагальнені показники для прийняття управлінських рішень.

У межах даної роботи розглядається побудова інформаційно-аналітичної системи на базі платформи Microsoft SQL Server із використанням сучасних технологій інтеграції даних, аналітичної обробки та формування звітності. Такий підхід дозволяє створити єдине інформаційне середовище, що забезпечує комплексний аналіз даних та підвищує ефективність управління.

Історія розвитку баз даних та аналітичних систем

Перші інформаційні системи використовували файловий підхід до зберігання даних, який передбачав збереження інформації у вигляді окремих файлів. Такий підхід мав низку недоліків, зокрема дублювання даних,

складність оновлення, низький рівень безпеки та відсутність механізмів забезпечення цілісності. Це ускладнювало масштабування систем і призводило до зростання кількості помилок у даних.

З розвитком обчислювальної техніки та зростанням обсягів інформації була запропонована концепція систем управління базами даних. Важливим етапом стало впровадження реляційної моделі даних, яка дозволила логічно структурувати інформацію у вигляді таблиць та визначити зв'язки між ними. Використання мови SQL забезпечило уніфікований доступ до даних і значно спростило процес їх обробки.

Однак реляційні бази даних здебільшого орієнтовані на оперативну обробку транзакцій, а не на складний аналітичний аналіз. Зі збільшенням обсягів історичних даних та ускладненням запитів виникла потреба у спеціалізованих аналітичних рішеннях. Так з'явилася концепція сховищ даних, основною метою яких є підтримка аналітичних процесів та бізнес-аналітики.

Сховища даних та їх роль в аналітичних системах

Сховище даних являє собою централізоване сховище інформації, яке формується на основі даних із різних операційних систем. На відміну від транзакційних баз даних, сховища орієнтовані на зберігання історичної інформації та виконання аналітичних запитів. Дані у сховищі зазвичай організовуються за багатовимірною моделлю, що полегшує їх аналіз.

Однією з ключових особливостей сховищ даних є використання процесів ETL (Extract, Transform, Load), які забезпечують вилучення даних із джерел, їх очищення, перетворення та завантаження у цільову структуру. Завдяки цьому забезпечується узгодженість, цілісність і висока якість даних, що є критично важливим для подальшого аналізу.

Olap-технології та багатовимірний аналіз даних

OLAP-технології (Online Analytical Processing) є основою сучасних аналітичних систем. Вони забезпечують можливість багатовимірного аналізу даних, що дозволяє розглядати інформацію з різних точок зору. Дані в OLAP-системах організовуються у вигляді кубів, які складаються з вимірів і показників.

Використання OLAP-кубів дає змогу виконувати операції агрегації, деталізації, фільтрації та порівняння даних. Це значно підвищує швидкість аналізу та робить роботу з великими обсягами інформації більш ефективною. Саме тому ці технології широко використовуються у сучасних інформаційно-аналітичних системах.

Платформа Microsoft SQL Server та її компоненти

Платформа Microsoft SQL Server є одним із найпоширеніших рішень для побудови корпоративних інформаційно-аналітичних систем. Вона забезпечує повний набір інструментів для зберігання, інтеграції, аналізу та візуалізації даних. Однією з ключових переваг цієї платформи є інтеграція всіх компонентів у єдине середовище.

SQL Server Integration Services використовується для реалізації ETL-процесів і дозволяє автоматизувати завантаження даних із різних джерел. SQL Server Analysis Services забезпечує створення OLAP-кубів і підтримує багатовимірний аналіз даних. SQL Server Reporting Services надає засоби для формування звітів і представлення результатів аналізу у зручному для користувача вигляді.

Розділ 1. Аналіз предметної області

"Оренда приміщень"

Основні сутності: Приміщення, Типи приміщень, Орендарі, Договори оренди, Рахунки, Платежі.

Специфічні вимоги:

• Мінімум 5 000 приміщень

• Мінімум 100 000 операцій оренди

• Історія за 5 років

Обов'язкові звіти:

1. Список орендарів з інформацією про оренду та борги
2. Стан приміщень на певну дату
3. Довідка орендаря про історію оренди та платежів
4. Аналіз завантаженості приміщень
5. Фінансовий звіт по орендарях

База даних призначена для зберігання та обробки інформації, пов’язаної з процесами оренди приміщень. Вона охоплює основні сутності предметної області та відображає взаємозв’язки між приміщеннями, орендарями, договорами оренди, рахунками та платежами. Структура бази даних дозволяє зберігати як поточні дані, так і історію орендних відносин за різні періоди часу.

Інформація про приміщення організована з урахуванням їх типів. Типи приміщень використовуються для класифікації об’єктів оренди за функціональним призначенням. Кожне приміщення має унікальний ідентифікатор, номер, площу та ознаку активності, що дає змогу враховувати як доступні, так і тимчасово неактивні приміщення.

Орендарі зберігаються як окрема сутність і містять базову ідентифікаційну та контактну інформацію. Це дозволяє пов’язувати одного

орендаря з кількома договорами оренди та відстежувати його взаємодію з системою впродовж часу. Такий підхід спрощує аналіз історії оренди та фінансових зобов’язань орендарів.

Договори оренди відображають факт передачі конкретного приміщення в оренду певному орендарю. Для кожного договору фіксується період дії та розмір щомісячної орендної плати. Наявність дат початку та завершення договору дозволяє визначати активні орендні відносини на будь-яку обрану дату.

Фінансовий облік реалізується за допомогою рахунків і платежів. Рахунки формуються на підставі договорів оренди та містять інформацію про дату виставлення, суму та статус оплати. Платежі зберігають дані про фактичне надходження коштів і пов’язуються з відповідними рахунками, що дає змогу контролювати повноту та своєчасність оплат.

Така структура бази даних забезпечує логічну цілісність інформації, підтримує аналіз орендних і фінансових процесів та створює основу для формування звітів щодо завантаженості приміщень, фінансових показників і діяльності орендарів.

Розділ 2. Проектування бази даних

База даних для заданої предметної області складатиметься з 6 таблиць:

Tenant(Орендар) містить дані про орендаря. Зберігає унікальний номер, повне ім’я, телефонний номер та електронну адресу

RoomType(Тип приміщення) зберігає дані про види приміщень для оренди. Містить унікальний ключ та назву типу.

Room(Приміщення) зберігає дані про приміщення які коли-небудь здавались в оренду. Містить унікальний системний ідентифікатор та окремий номер, тип, площу та статус активності.

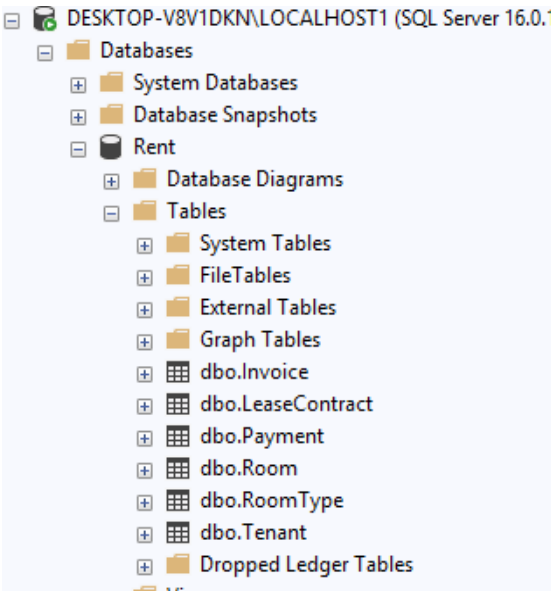
LeaseContract(Контракт) зберігає історію договорів про оренду. Містить унікальний номер договору, приміщення та орендаря, дату початку та кінця дійсності контракту та ціну.

Invoice(Рахунок) зберігає історію всіх рахунків. Містить унікальний номер рахунка та договору, дату, суму та статус рахунку (оплачено чи ні).

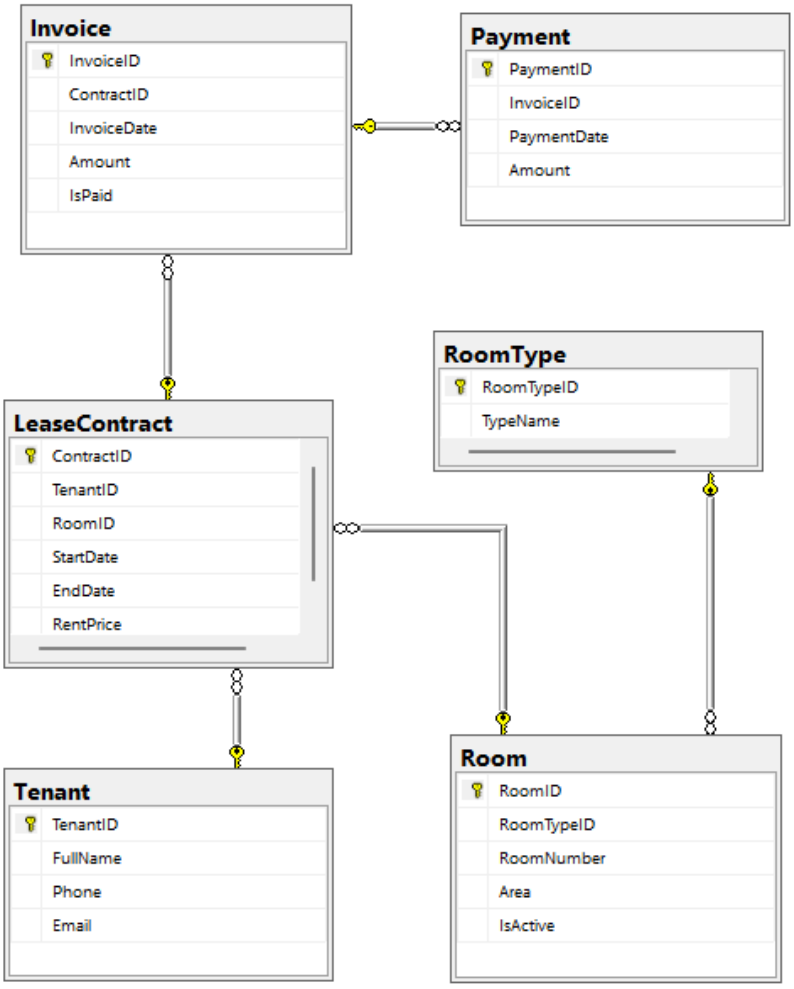
Payment(Оплата) зберігає історію всіх платежів. Містить унікальний номер платежів та рахунків, день платежу та суму.

Створення бази даних

У програмі SQL Server Management Studio за допомогою SQL-скриптів створено базу даних з назвою Rent та необхідні таблиці. Таблиці одразу створені з ключами та зв’язками між ними. У результаті сформовано повноцінну структуру бази даних для обліку оренди приміщень:

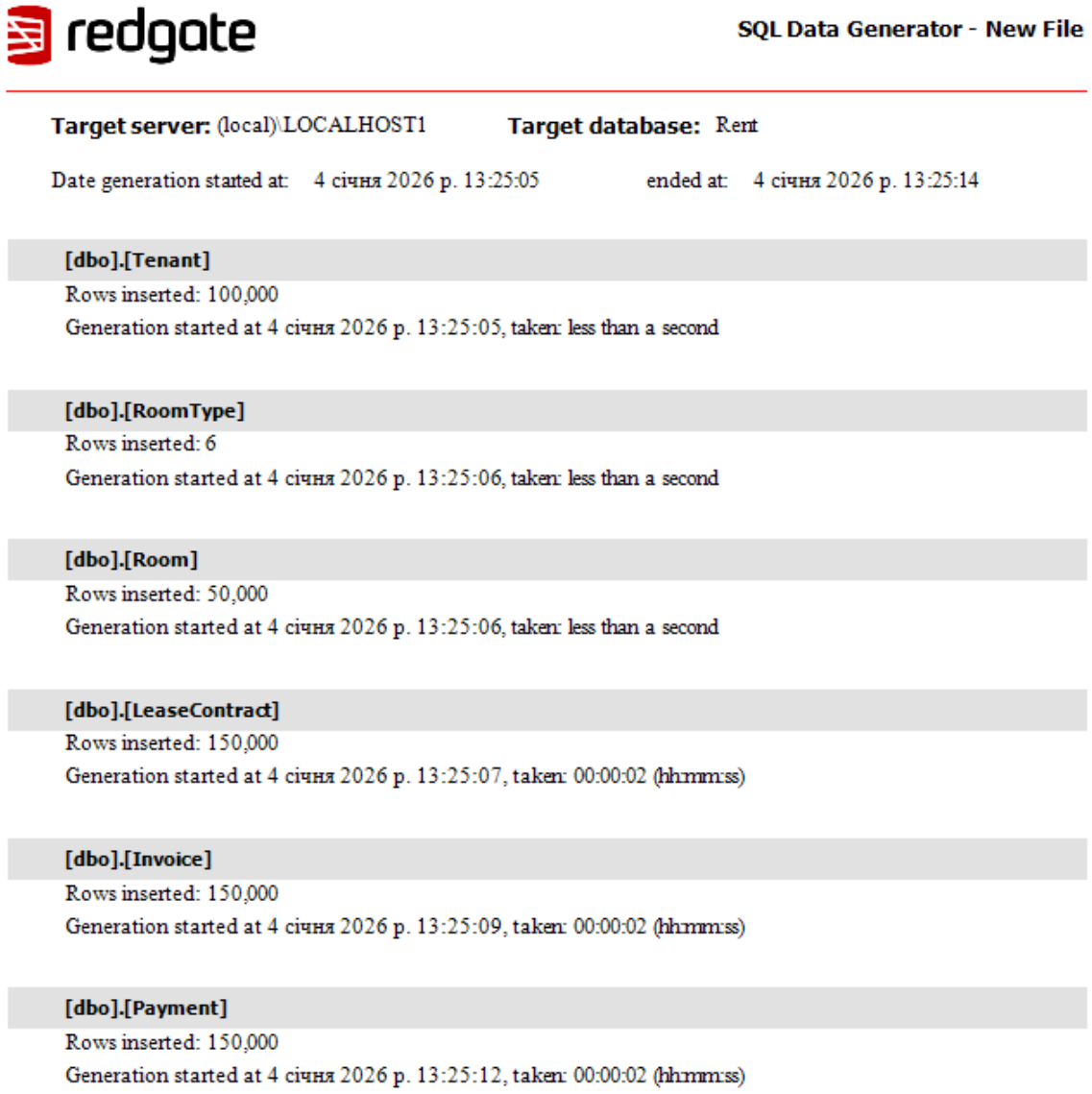


Діаграма бази даних:



Генерація даних

За допомогою програми Red Gate SQL Data Generator таблиці заповнені реалістичними даними. На фото-звіті можна побачити скільки рядків в кожній таблиці.



redgate

SQL Data Generator - New File

Target server: (local)\LOCALHOST1 **Target database:** Rent

Date generation started at: 4 січня 2026 р. 13:25:05 ended at: 4 січня 2026 р. 13:25:14

[dbo].[Tenant]
Rows inserted: 100,000
Generation started at 4 січня 2026 р. 13:25:05, taken: less than a second

[dbo].[RoomType]
Rows inserted: 6
Generation started at 4 січня 2026 р. 13:25:06, taken: less than a second

[dbo].[Room]
Rows inserted: 50,000
Generation started at 4 січня 2026 р. 13:25:06, taken: less than a second

[dbo].[LeaseContract]
Rows inserted: 150,000
Generation started at 4 січня 2026 р. 13:25:07, taken: 00:00:02 (hh:mm:ss)

[dbo].[Invoice]
Rows inserted: 150,000
Generation started at 4 січня 2026 р. 13:25:09, taken: 00:00:02 (hh:mm:ss)

[dbo].[Payment]
Rows inserted: 150,000
Generation started at 4 січня 2026 р. 13:25:12, taken: 00:00:02 (hh:mm:ss)

За допомогою скриптів, зроблено запити на кількість рядків в таблицях, щоб пепервірити чи дані згенеровані та чи вони коректні:

```
1  SELECT COUNT(*) AS TotalRows
2  FROM dbo.Tenant;
```

100 % — No issues found

Results | Messages

	TotalRows
1	100000

```
1  SELECT COUNT(*) AS TotalRows
2  FROM dbo.RoomType;
```

100 % — No issues found

Results | Messages

	TotalRows
1	6

```
1  SELECT COUNT(*) AS TotalRows
2  FROM dbo.Room;
```

100 % — No issues found

Results | Messages

	TotalRows
1	50000

```
1  SELECT COUNT(*) AS TotalRows
2  FROM dbo.LeaseContract;
```

100 % — No issues found

Results | Messages

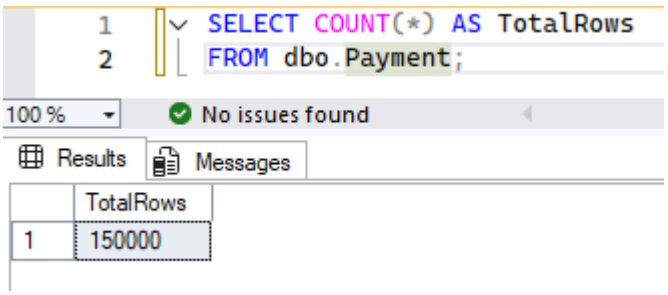
	TotalRows
1	150000

```
1  SELECT COUNT(*) AS TotalRows
2  FROM dbo.Invoice;
```

100 % — No issues found

Results | Messages

	TotalRows
1	150000



Для коректності даних, за допомогою скриптових запитів, було оновленно конкретний рядок з даними. А саме: в таблиці з контрактами оновлено кінець дії контракту так, щоб уникнути негативного значення терміну дії.

Розділ 3. Реалізація ETL-процесів

Створення Data Warehouse

Для ефективного зберігання та аналітичної обробки даних оренди приміщень було спроектовано сховище даних (DW) із використанням зіркової схеми (Star Schema). Така структура дозволяє швидко виконувати аналітичні запити та формувати звіти, оскільки таблиці фактів з'єднані з таблицями вимірів за допомогою ключів, що забезпечує логічну цілісність даних та простоту агрегацій. Сховище підтримує історію змін у даних за допомогою реалізації SCD (Slowly Changing Dimensions) Type 1, що дозволяє оновлювати характеристики вимірів без збереження повної історії попередніх значень.

База даних DW була створена за допомогою SQL-скриптів і отримала назву RentDW. Вона включає декілька таблиць вимірів та фактів, що відображають основні бізнес-процеси оренди приміщень.

Таблиці вимірів:

DimTenant (Вимір «Орендар») містить дані про орендарів, включаючи унікальний технічний ключ, бізнес-ключ, повне ім'я, телефонний номер та електронний адрес, дати початку та завершення орендних відносин, а також ознаку поточного статусу орендаря.
DimRoom (Вимір «Приміщення») містить інформацію про приміщення, включаючи технічний ключ, бізнес-ключ, номер приміщення, тип та площу.
DimRoomType (Вимір «Тип приміщення») містить класифікацію приміщень за типами, з технічним та бізнес-ключем і назвою типу.
DimDate (Вимір «Дата») використовується для аналітики по часових показниках. Містить ключ, повну дату, рік, місяць та день. Таблиця була заповнена на основі періоду з 2019 по 2024 рік за допомогою sql-запиту.

DimContract (Вимір «Договір оренди») містить інформацію про договори

оренди, включаючи ключі, дати початку та завершення дії договору та орендну плату.
DimPaymentStatus (Вимір «Статус оплати») містить статуси оплати рахунків: «Оплачено» або «Не оплачено».

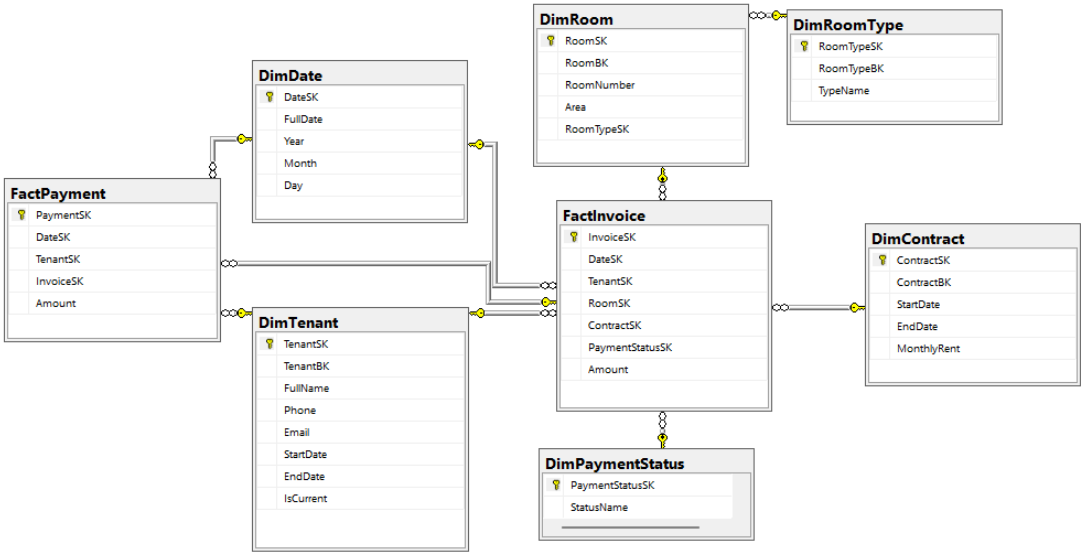
Таблиці фактів:

FactInvoice (Факт «Рахунок») містить інформацію про рахунки за оренду, включаючи посилання на дату, орендаря, приміщення, договір і статус оплати, а також суму рахунку.

FactPayment (Факт «Платіж») містить фактичні платежі, пов’язані з рахунками, включаючи дату, орендаря, суму та посилання на відповідний рахунок.

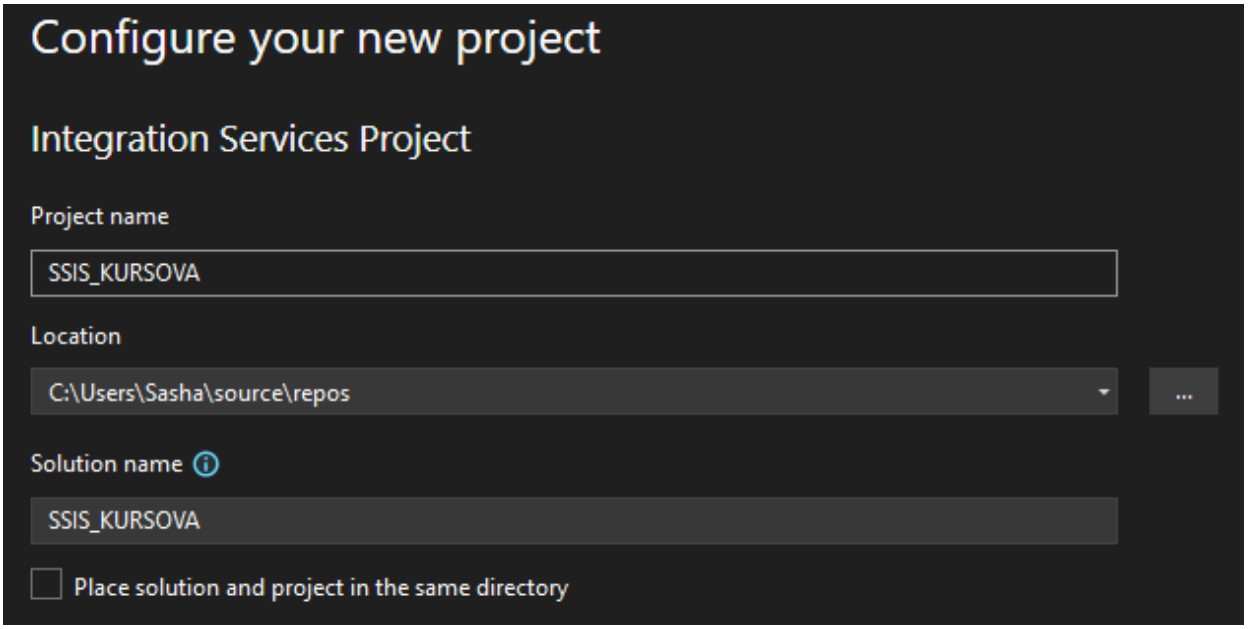
Така структура DW дозволяє виконувати аналітичні запити щодо фінансової діяльності орендарів, завантаженості приміщень, відстеження боргів і платежів, а також формувати різноманітні звіти для управлінських рішень. Використання зіркової схеми забезпечує високу продуктивність агрегацій та спрощує інтеграцію з OLAP-кубами та іншими аналітичними інструментами.

Діаграма сховища даних:

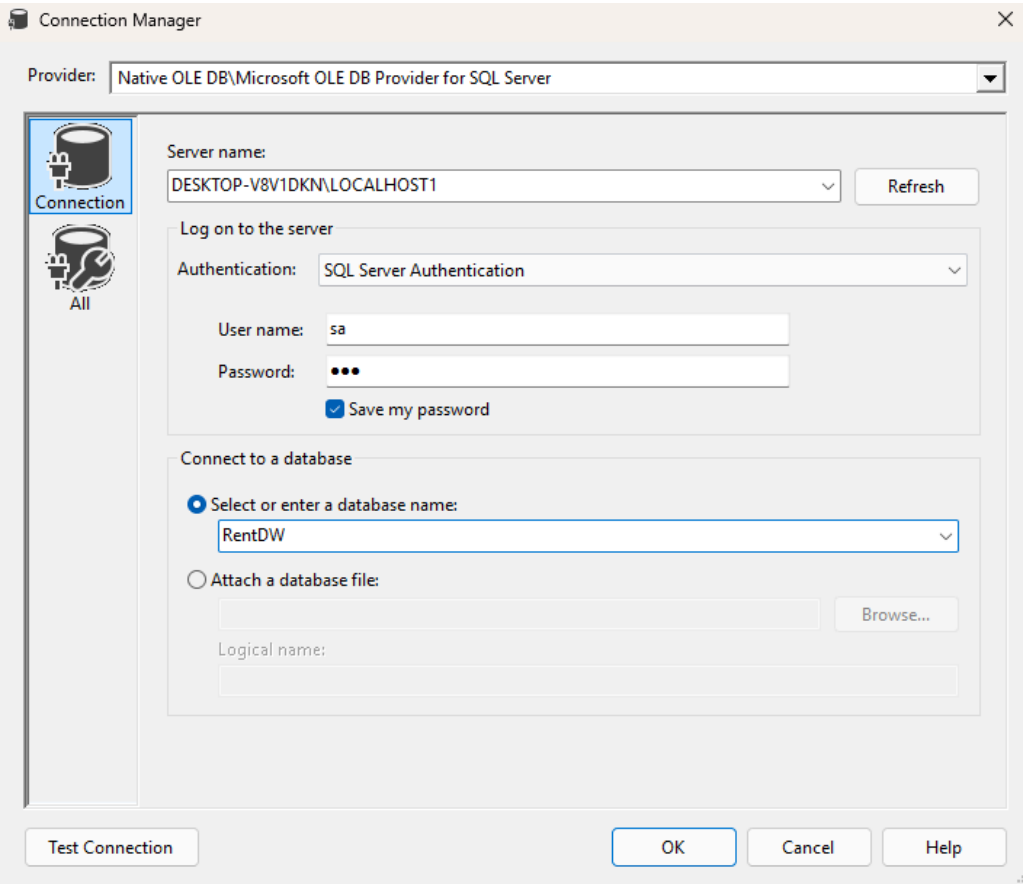


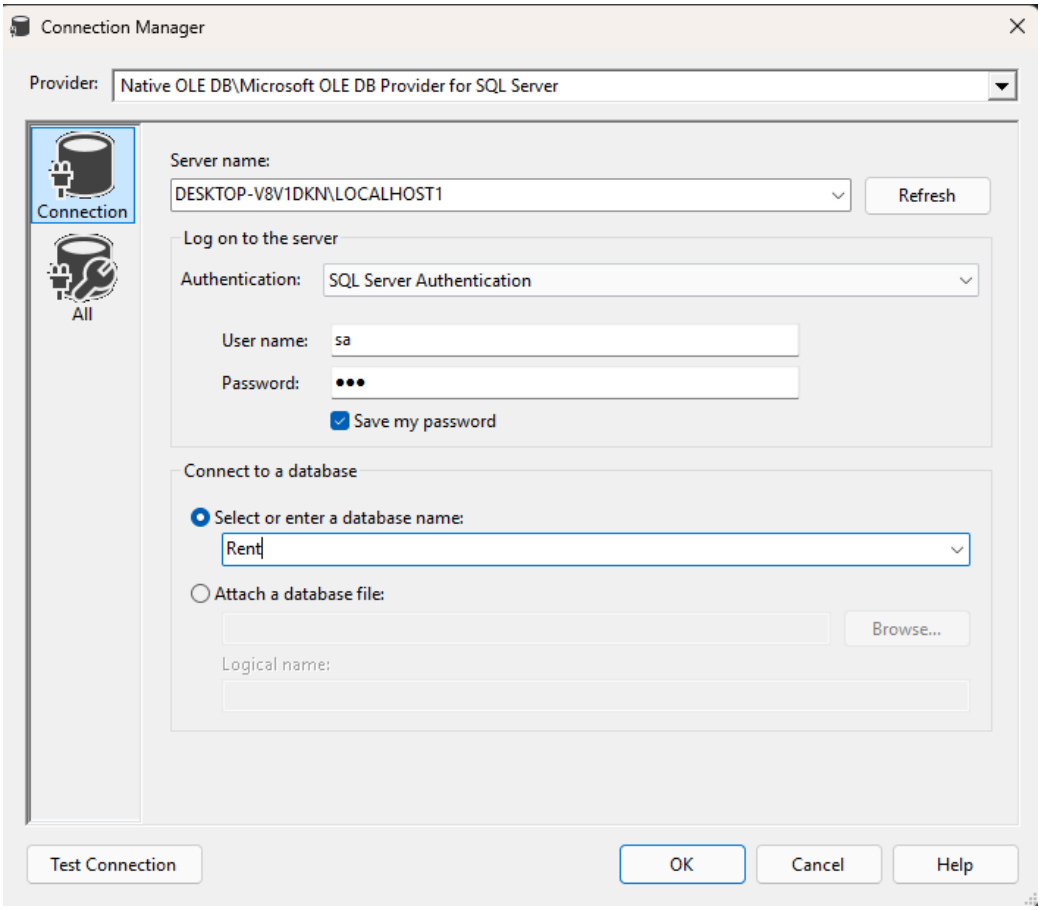
Створення проекту Integration Services

В середовищі Visual Studio, створено новий SSIS-проєкт під назвою SSIS_KURSOVA.



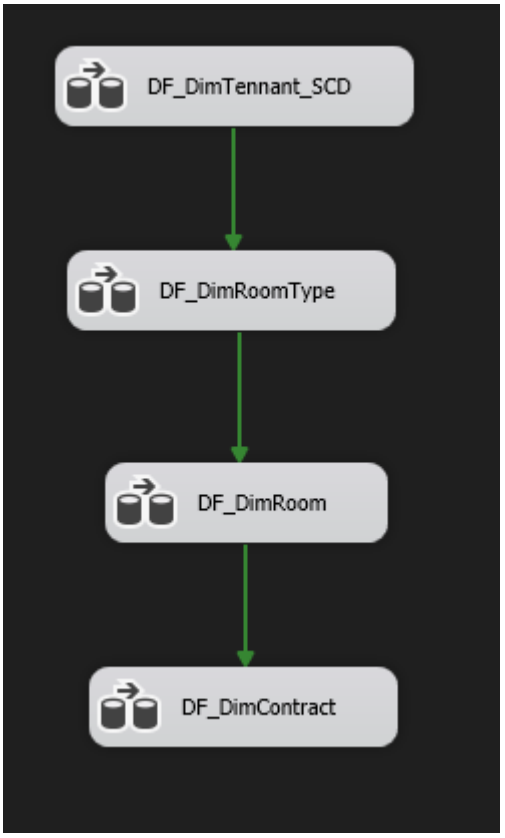
Підключено до бази даних та сховища даних за допомогою Conection Manager:


Connection Manager
Provider:
Native OLE DB\Microsoft OLE DB Provider for SQL Server
Connection
All
Server name:
DESKTOP-V8V1DKN\LOCALHOST1
Refresh
Log on to the server
Authentication:
SQL Server Authentication
User name:
sa
Password:
Save my password
Connect to a database
Select or enter a database name:
RentDW
Attach a database file:
Browse...
Logical name:
Test Connection
OK
Cancel
Help

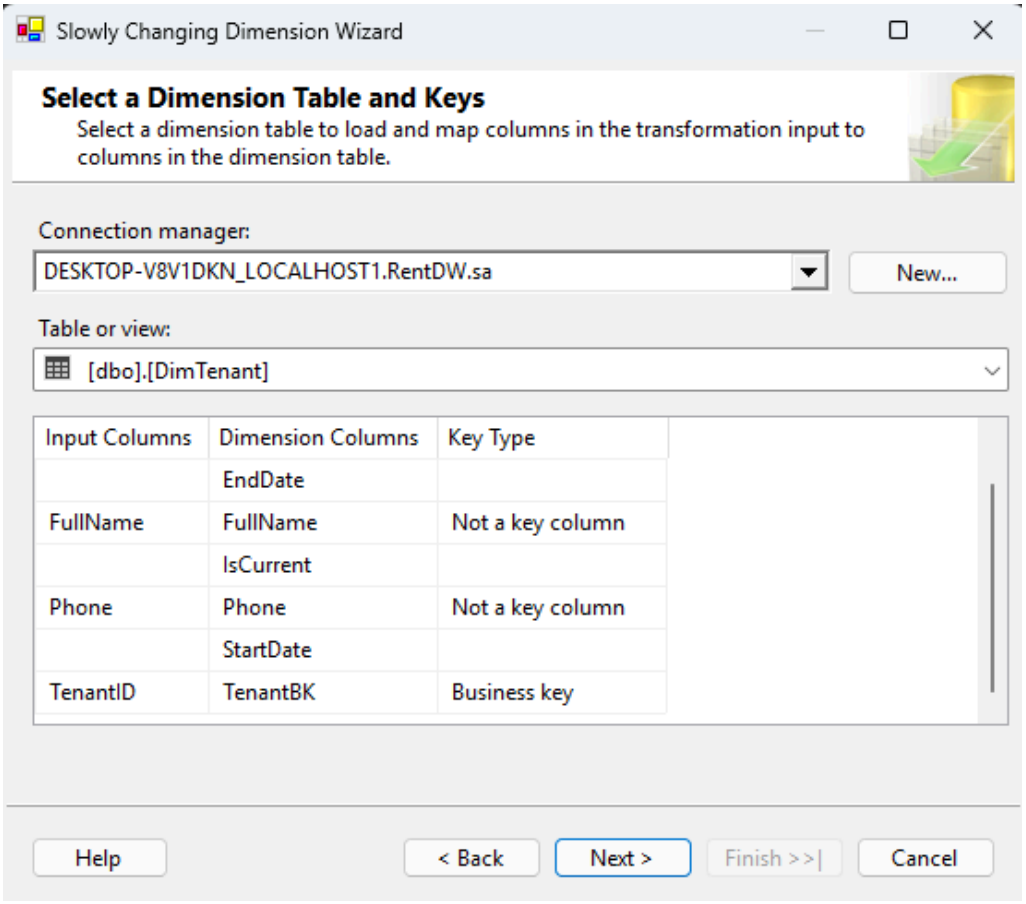

Connection Manager
Provider:
Native OLE DB\Microsoft OLE DB Provider for SQL Server
Connection
All
Server name:
DESKTOP-V8V1DKN\LOCALHOST1
Refresh
Log on to the server
Authentication:
SQL Server Authentication
User name:
sa
Password:
Save my password
Connect to a database
Select or enter a database name:
Rent
Attach a database file:
Browse...
Logical name:
Test Connection
OK
Cancel
Help

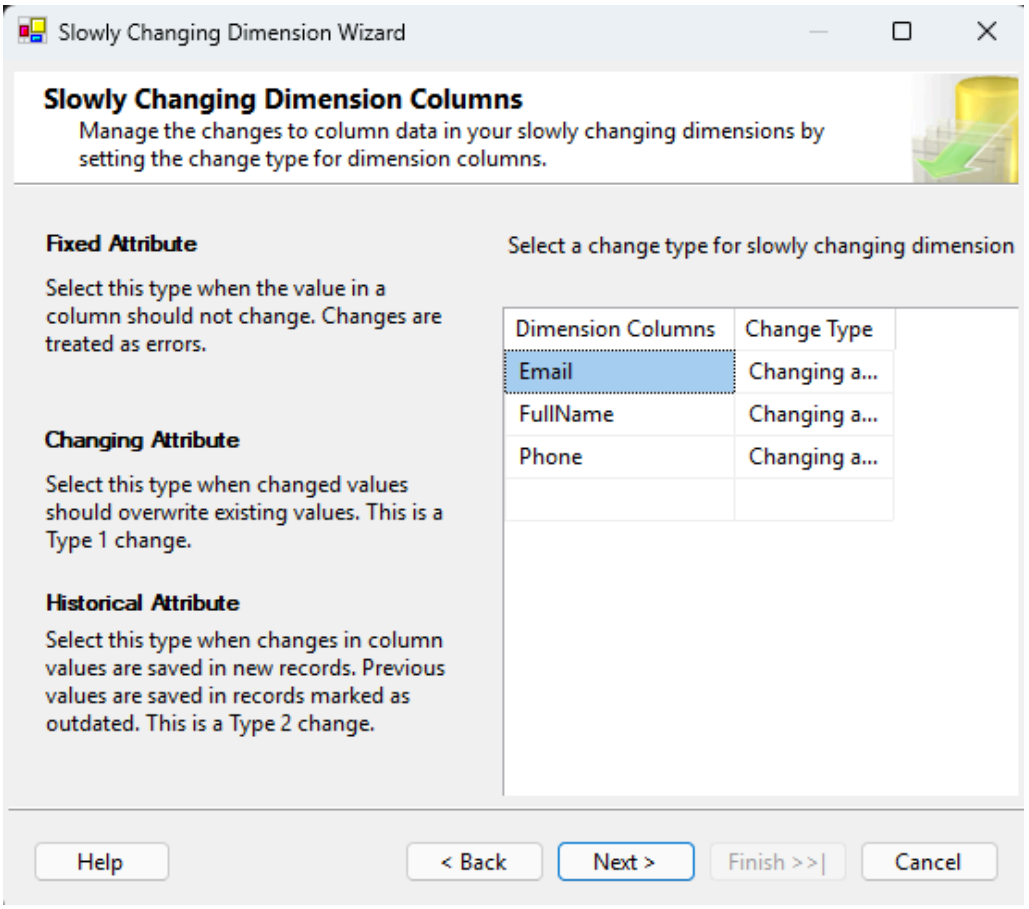
В проєкті створено два пакети: Dim.dtsx та Fact.dtsx.

В першому пакеті для вимірів створено 4 Data Flow Task блоки для таких вимірів: DimTenant, DimRoomType, DimRoom та DimContract. Проведено зв’язки між блоками:



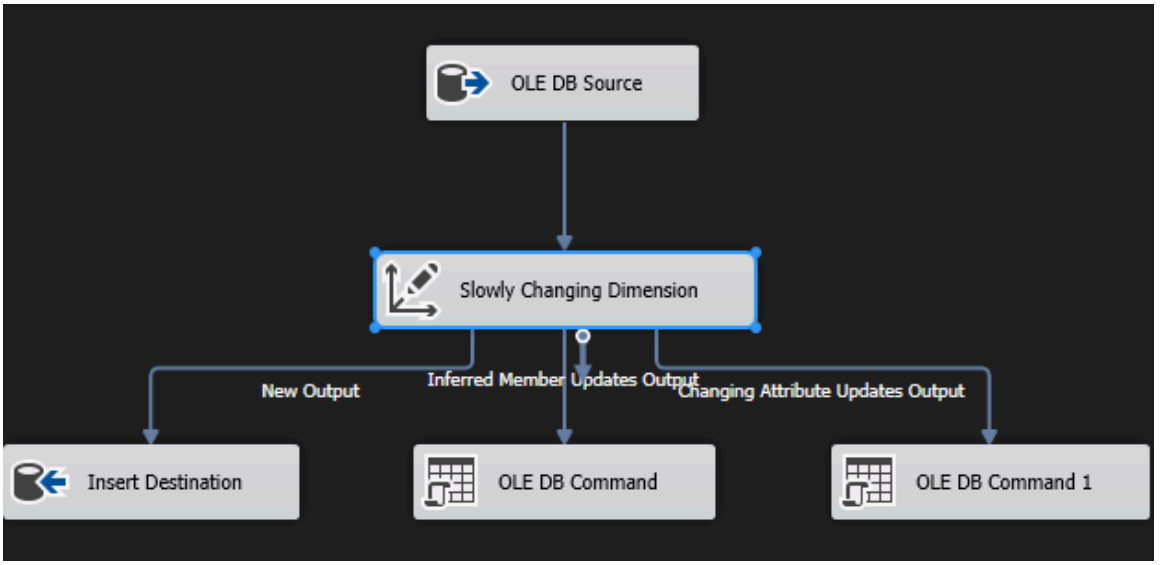
В першому Data Flow блоці створено OLE DB Source та обрано таблицю Tenant з бази даних Rent. Далі, додано компонент Slowly Changing Dimension, в якому унікальний номер орендаря обрано як бізнес ключ. Решту полів таблиці обрано для SCD Type-1, що дозволяє оновлювати атрибути без збереження історії попередніх значень.


Slowly Changing Dimension Wizard
Select a Dimension Table and Keys
Select a dimension table to load and map columns in the transformation input to columns in the dimension table.
Connection manager:
DESKTOP-V8V1DKN_LOCALHOST1.RentDW.sa
New...
Table or view:
[dbo].[DimTenant]
Input Columns
Dimension Columns
Key Type
EndDate
FullName
FullName
Not a key column
IsCurrent
Phone
Phone
Not a key column
StartDate
TenantID
TenantBK
Business key
Help
< Back
Next >
Finish >>|
Cancel

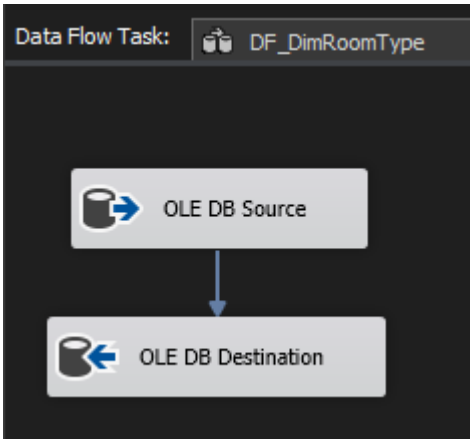

Slowly Changing Dimension Wizard
Slowly Changing Dimension Columns
Manage the changes to column data in your slowly changing dimensions by setting the change type for dimension columns.
Fixed Attribute
Select this type when the value in a column should not change. Changes are treated as errors.
Changing Attribute
Select this type when changed values should overwrite existing values. This is a Type 1 change.
Historical Attribute
Select this type when changes in column values are saved in new records. Previous values are saved in records marked as outdated. This is a Type 2 change.
Select a change type for slowly changing dimension
Dimension Columns
Change Type
Email
Changing a...
FullName
Changing a...
Phone
Changing a...
Help
< Back
Next >
Finish >>|
Cancel

В результаті створення компоненти SCD було автоматично додано прив’язану до неї OLE DB Destination, в якій, після цього, було обрано таблицю виміру орендаря (DimTenant).

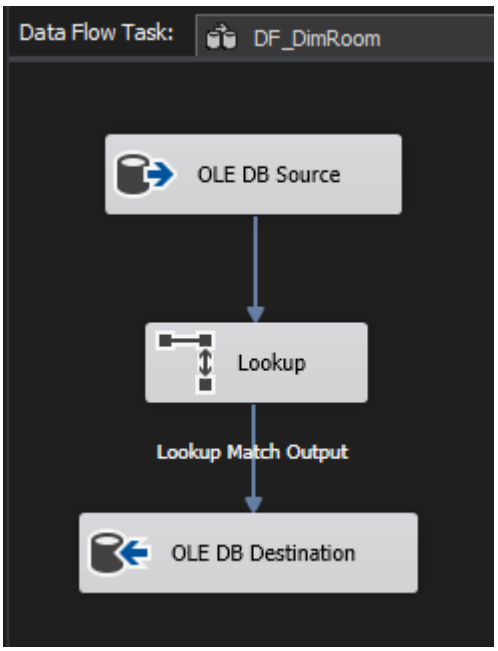
В результаті, в DF_DimTenant_SCD така структура:



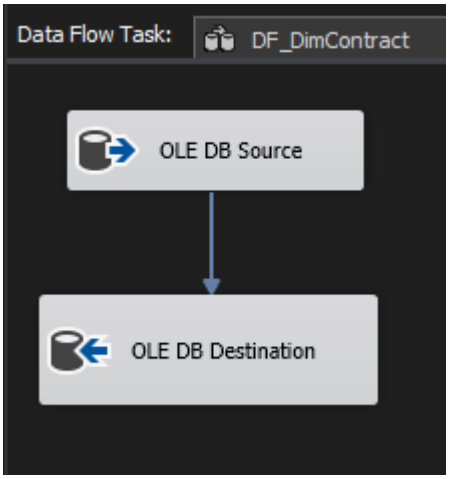
В Data Flow DF_DimRoomType створено компоненти OLE DB Source та Destination та обрано таблицю типів приміщення як джерело, а таблицю виміру як призначення:



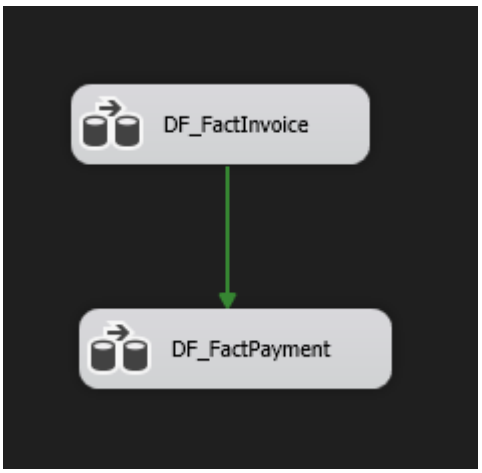
Аналогічно для DF_DimRoom, але з додаванням Lookup для зіставлення даних між таблицями вимірів приміщень та їх типів.



Для останнього Data Flow обрано таблицю контрактів як джерело даних та таблицю виміру як призначення.



В другому пакеті фактів створено два Data Flow Tasks для таблиць FactInvoice та FactPayment. Також створено зв’язки між ними:



Всередині першого Data Flow створено OLE DB Source з sql-скриптом як джерело даних, який посилається на потрібні для таблиці FactInvoice поля:

```
SELECT
    i.InvoiceID,
    i.InvoiceDate,
    i.Amount,
    i.IsPaid,
    c.ContractID,
    c.TenantID,
    c.RoomID
FROM Invoice i
JOIN LeaseContract c ON i.ContractID = c.ContractID;
```

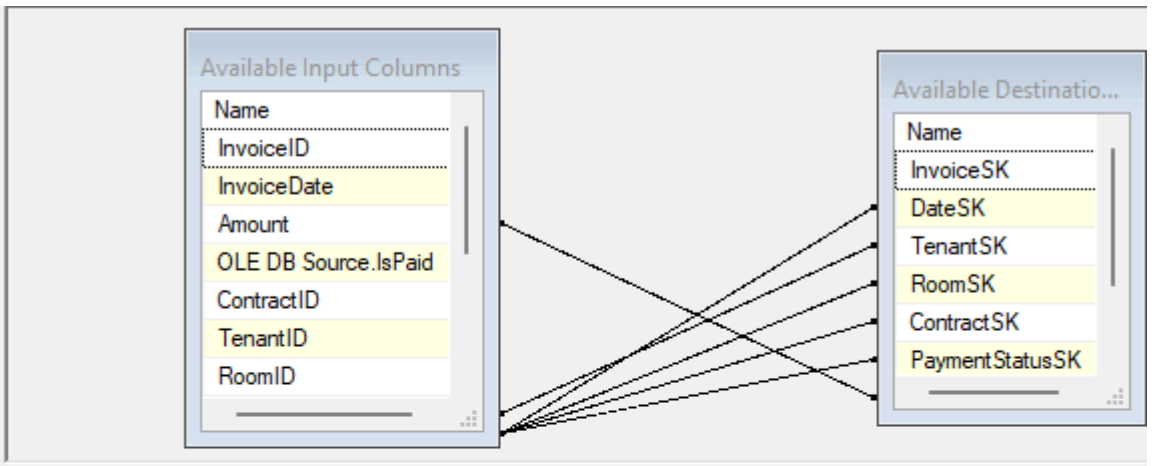
Також обрано блоки Lookup для зіставлення даних з таблицями вимірів (DimTenant, DimRoom, DimContract, DimDate) та, для коректності даних для виміру з датами та статусом оплати, було створено дві Derived Column, в яких додано наступні поля:

DateSK=(YEAR((DT_DBTIMESTAMP)InvoiceDate)*10000+MONTH((DT_DBTIMESTAMP)InvoiceDate)*100+DAY((DT_DBTIMESTAMP)InvoiceDate))

IsPaid=IsPaid == TRUE ? 1 : 2

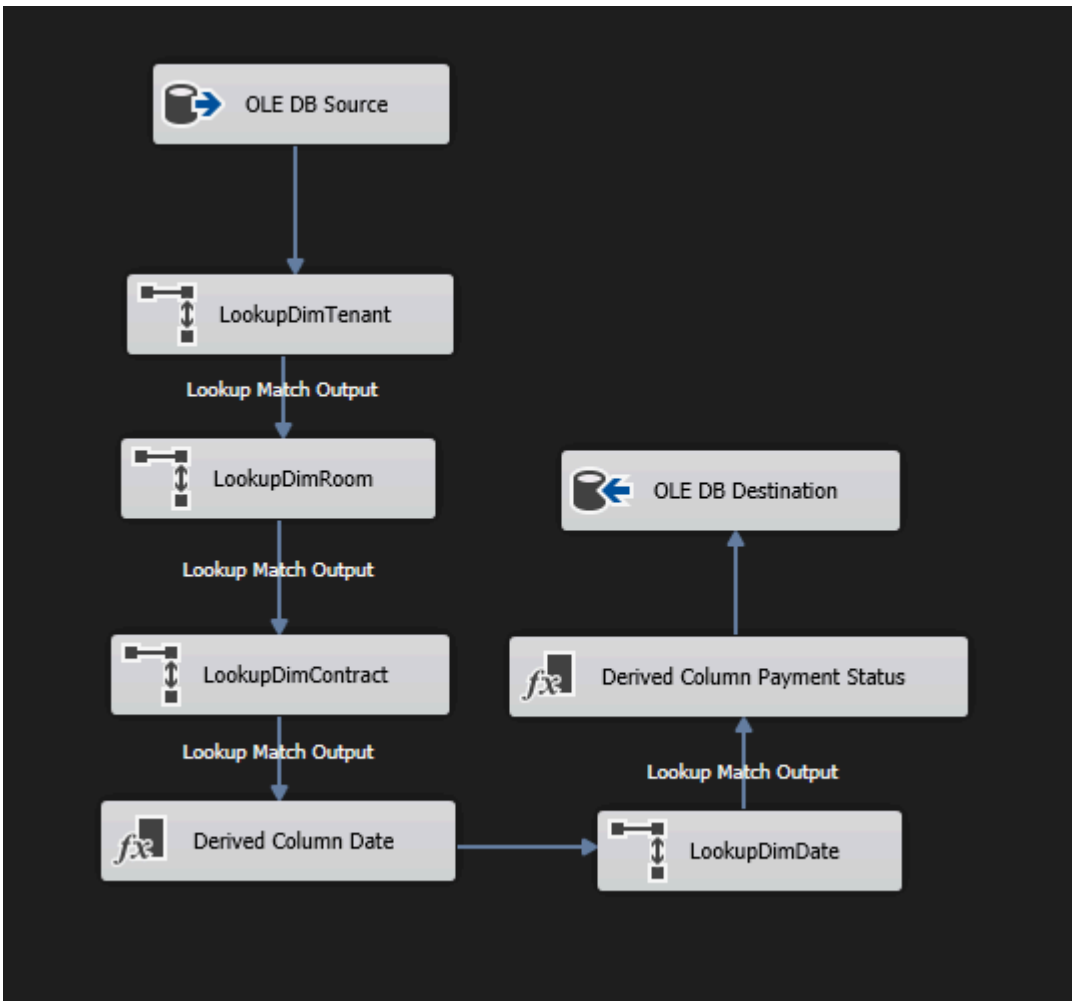
В OLE DB Destanation обрано таблицю FactInvoice.

В результаті вийшли такі зв’язки:



Input Column	Destination Column
<ignore>	InvoiceSK
LookupDimDate.DateSK	DateSK
TenantSK	TenantSK
RoomSK	RoomSK
ContractSK	ContractSK
Derived Column Payment Status.IsPaid	PaymentStatusSK
Amount	Amount

Структура:



Для наступного Data Flow створено OLE DB Source з sql-скриптом як джерело даних, який посилається на потрібні для таблиці FactPayment поля:

```
SELECT
   p.PaymentID,
   p.InvoiceID,
   p.PaymentDate,
   p.Amount,
   i.ContractID,
   c.TenantID
FROM Payment p
JOIN Invoice i ON p.InvoiceID = i.InvoiceID
JOIN LeaseContract c ON i.ContractID = c.ContractID;
```

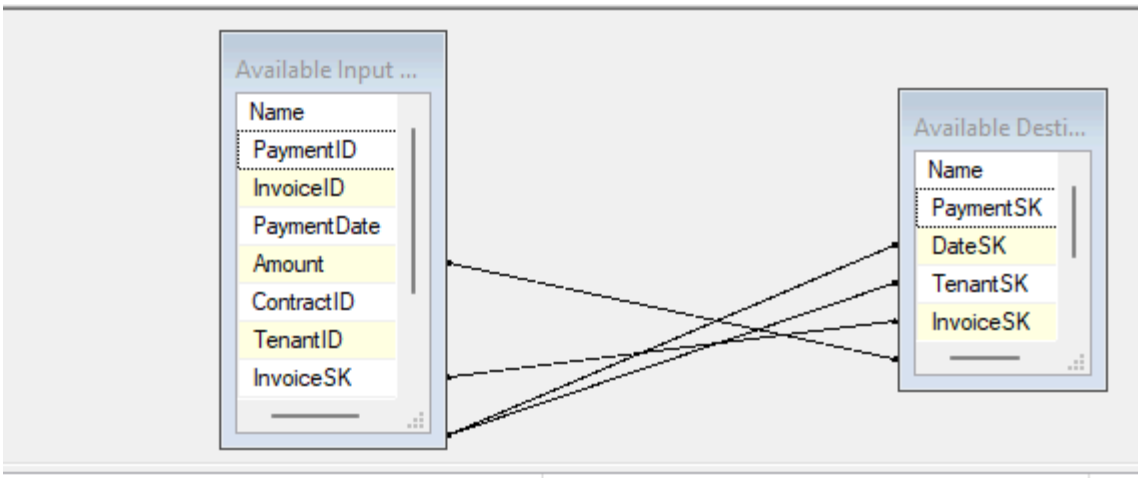
Також обрано блоки Lookup для зіставлення даних з таблицями (DimTenant, DimDate, FactInvoice) та, для коректності даних для виміру з датами та статусом оплати, було створено Derived Column, в якій було додано наступні поля:

додано наступні поля:

DateSK=(YEAR((DT_DBTIMESTAMP)InvoiceDate)*10000+MONTH((DT_DB TIMESTAMP)InvoiceDate)*100+DAY((DT_DBTIMESTAMP)InvoiceDate))

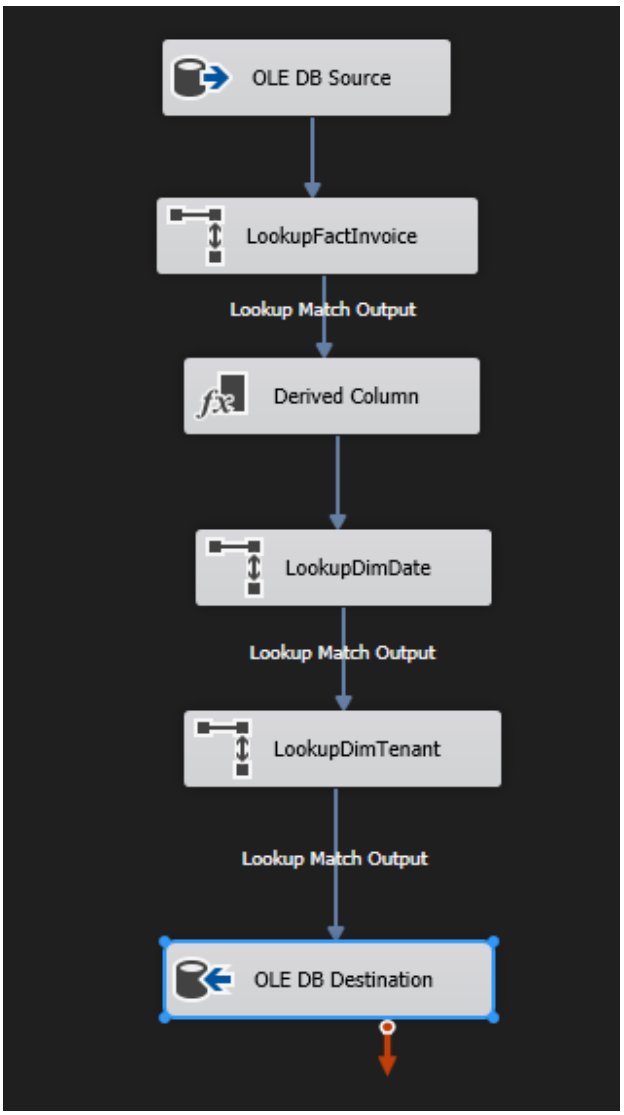
В OLE DB Destanation обрано таблицю FactPayment.

В результаті вийшли такі зв’язки:

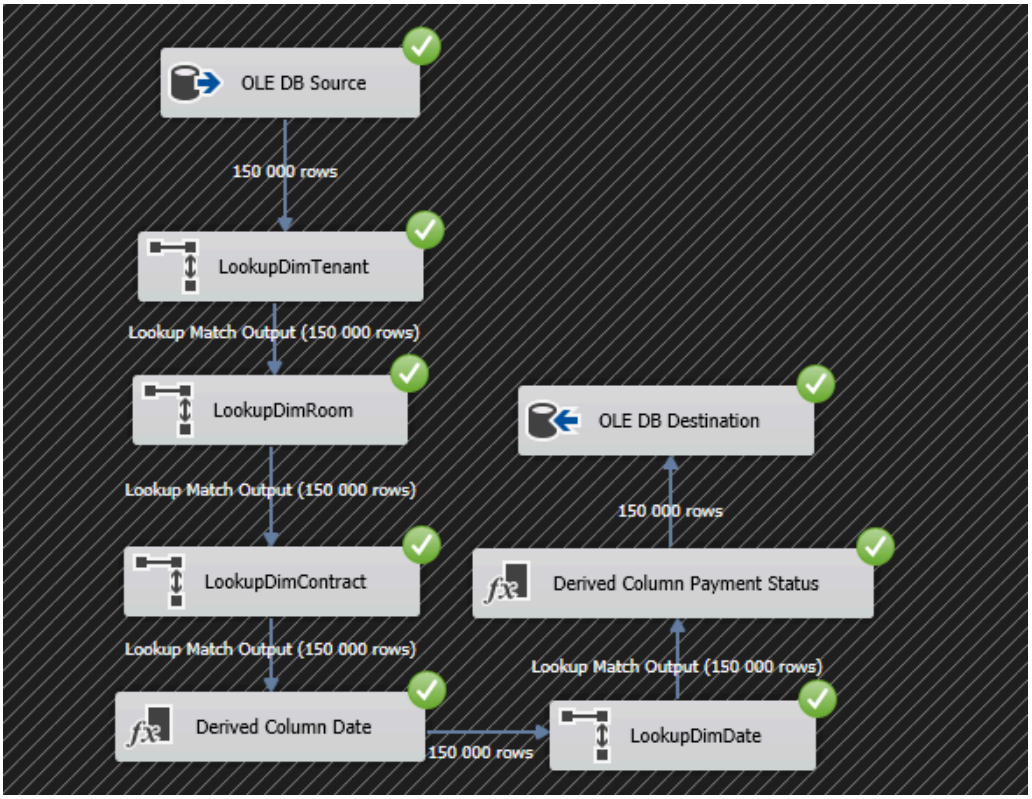


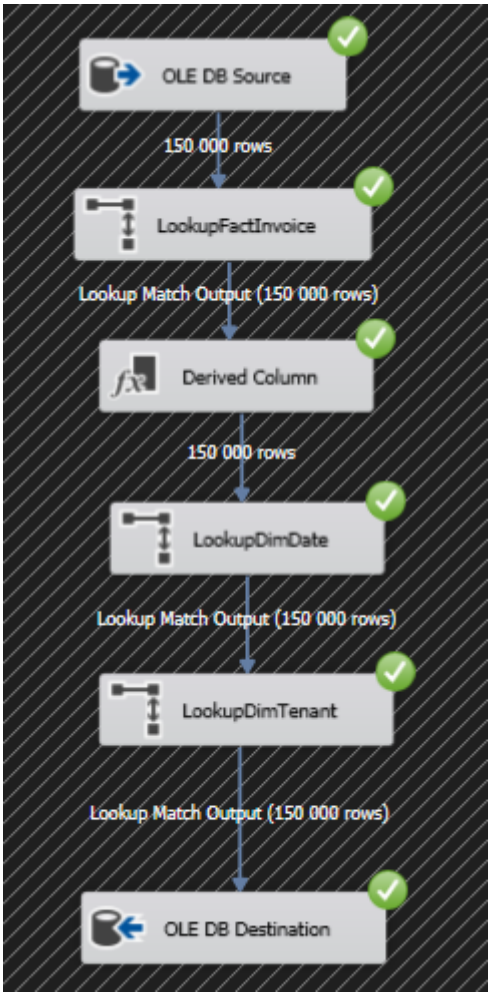
Input Column	Destination Column
<ignore>	PaymentSK
LookupDimDate.DateSK	DateSK
TenantSK	TenantSK
InvoiceSK	InvoiceSK
Amount	Amount

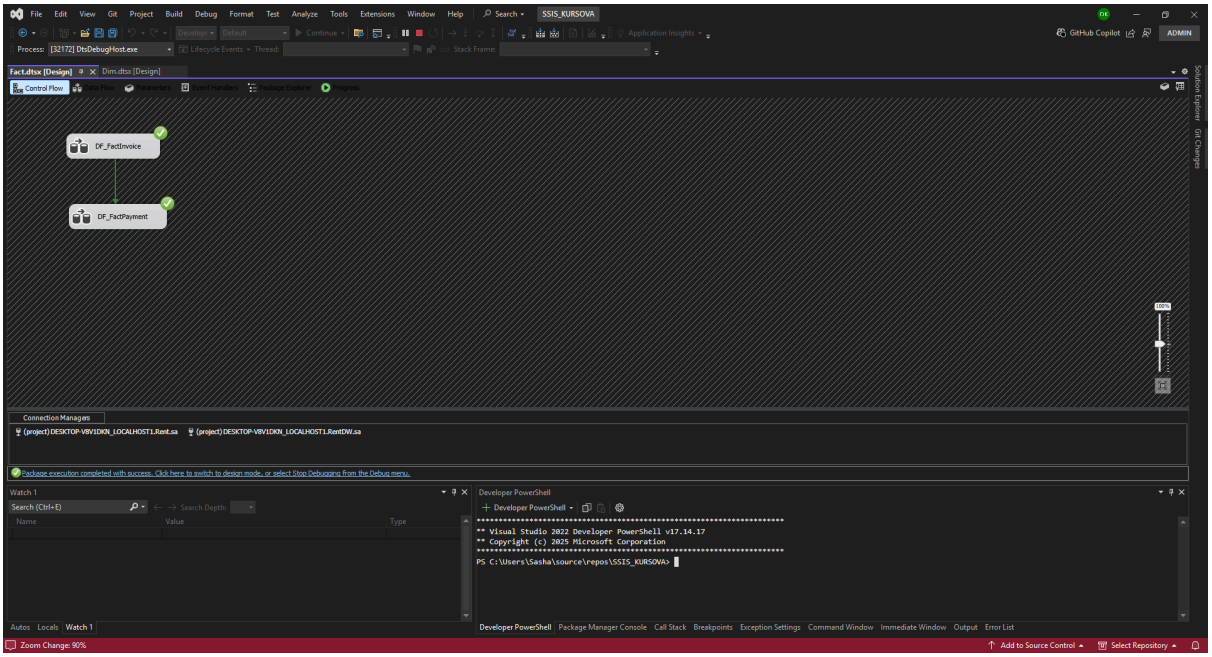
Та структура:



Запуск SSIS-проєкту:


OLE DB Source
150 000 rows
LookupDimTenant
Lookup Match Output (150 000 rows)
LookupDimRoom
Lookup Match Output (150 000 rows)
LookupDimContract
Lookup Match Output (150 000 rows)
Derived Column Date
150 000 rows
LookupDimDate
Lookup Match Output (150 000 rows)
Derived Column Payment Status
150 000 rows
OLE DB Destination


OLE DB Source
150 000 rows
LookupFactInvoice
Lookup Match Output (150 000 rows)
Derived Column
150 000 rows
LookupDimDate
Lookup Match Output (150 000 rows)
LookupDimTenant
Lookup Match Output (150 000 rows)
OLE DB Destination

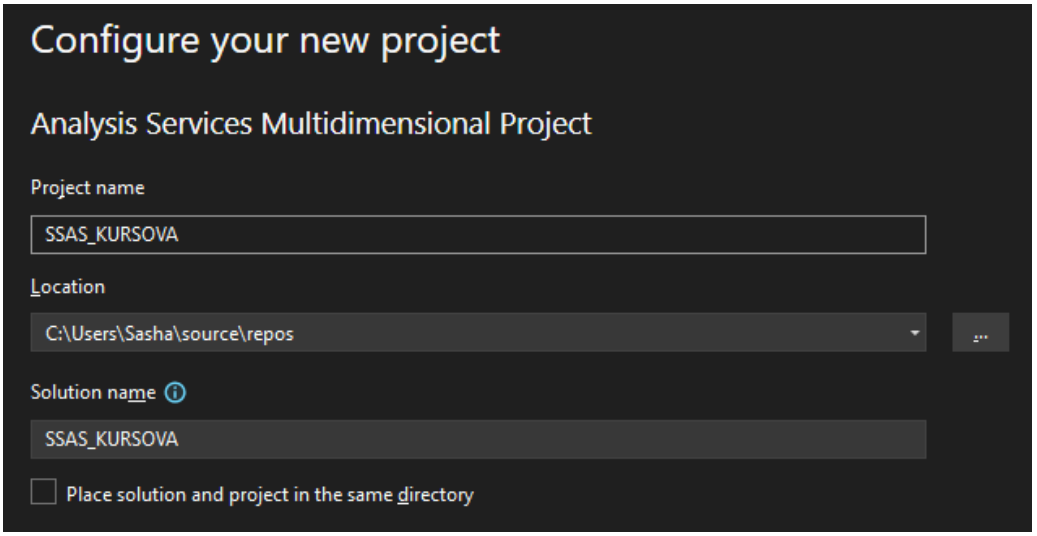


Проєкт запущено з успіхом та без помилок.

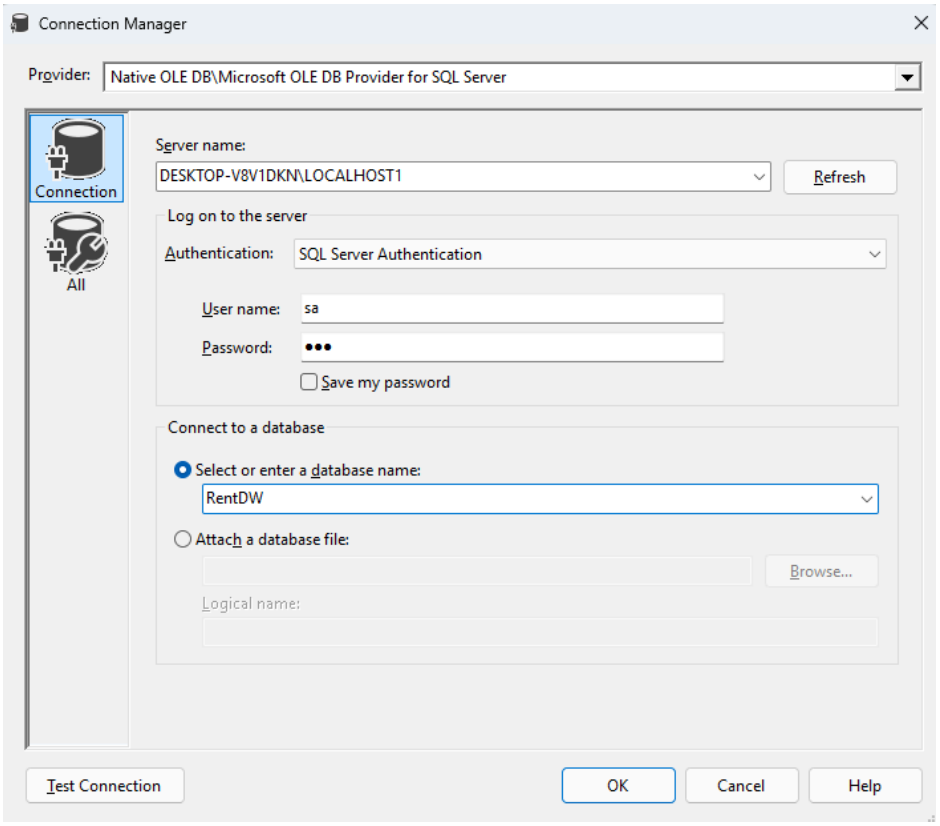
Розділ 4. Побудова OLAP-куба та аналітичні звіти

Побудова OLAP-куба

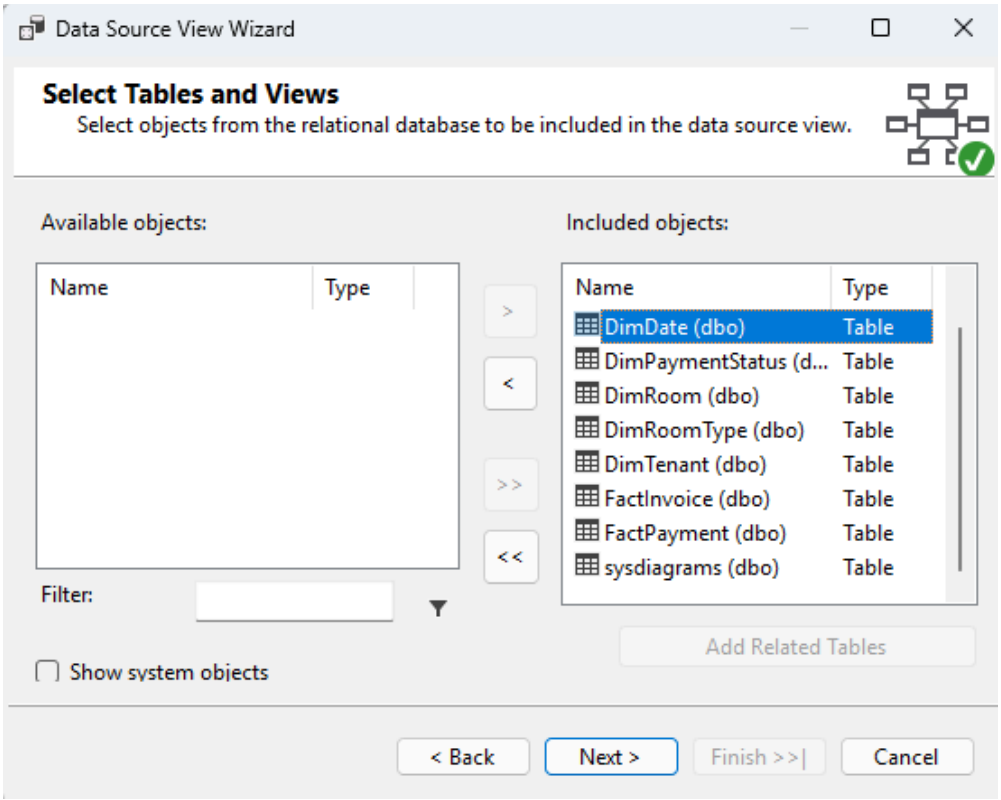
Спочатку створимо Analysis Services проєкт в середовищі Visual Studio та назвемо його SSAS_KURSOVA:



В Data Sources під'єднаємо сховище даних. Обираємо New Data Source та додаємо RentDW:

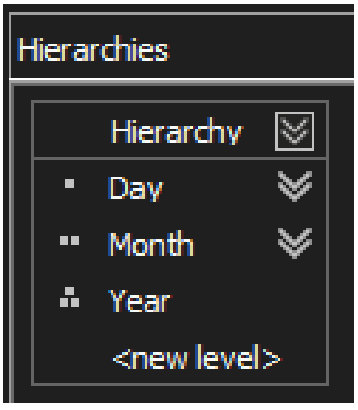


У вкладці Data Views обираємо New Data View та додаємо всі таблиці:



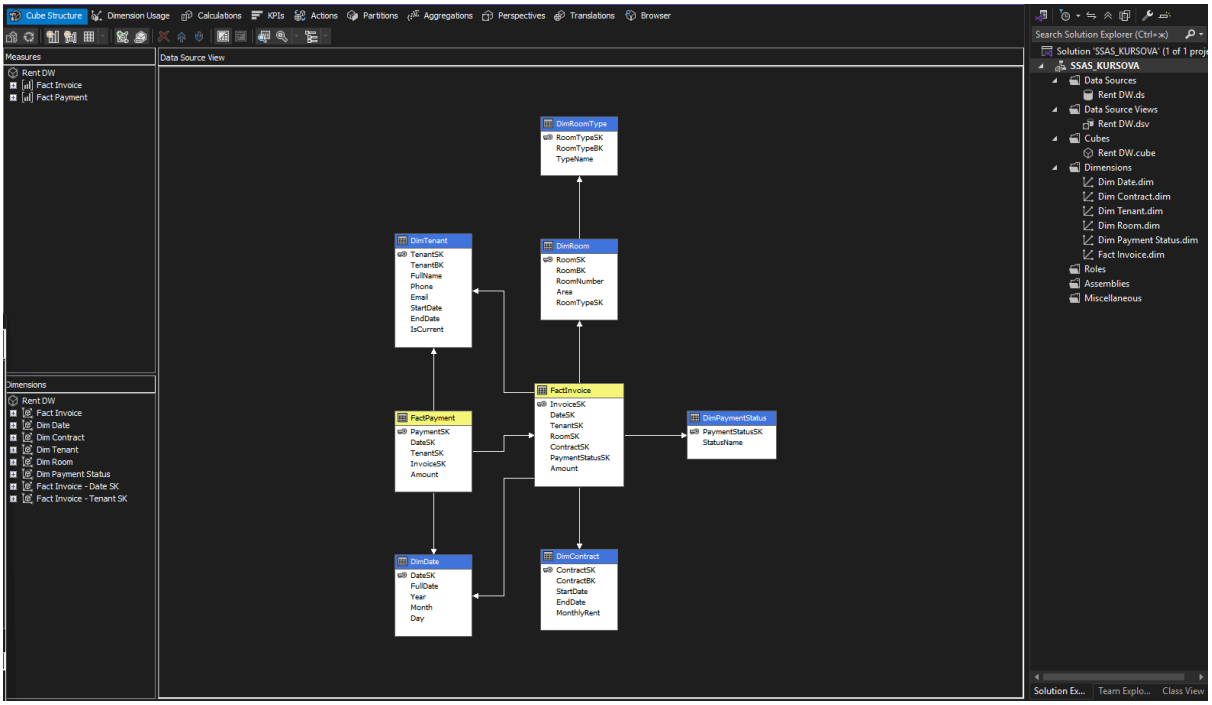
Після чого, в вкладці Dimensions, обираємо New Dimension та додаємо таблиці вимірів.

Для виміру дати налаштовуємо ієрархію:

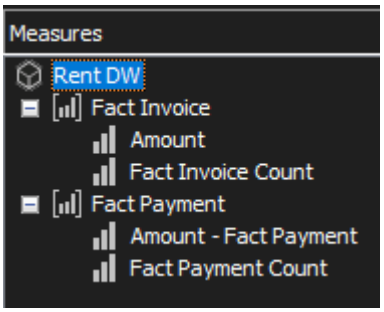


Далі створюємо OLAP-куб у середовищі SQL Server Analysis Services (SSAS). Для цього обраємо опцію New Cube, після чого в налаштуваннях вибираємо “Використати існуючі таблиці”. Ставимо галочку біля таблиць фактів, які слугуватимуть групами мір (Measures) у кубі. Після цього автоматично сформувалася схема куба, де таблиці фактів пов’язані з

відповідними таблицями вимірів:



В вкладці Measures було автоматично створено базові міри, а сама рахування кількості рядків в таблицях фактів та суму значень плати за оренду.



В вкладці Calcualtions створимо обчислювальні елементи за допомогою MDX-формул:

[TotalInvoice] = SUM([Measures].[Amount])

[TotalPayment]=SUM([Measures].[Amount - Fact Payment])

[Debt]=[TotalInvoice] - [TotalPayment]

[PaymentStatusText]=IIF([DimPaymentStatus].[PaymentStatusSK].CURRENTMEMBER.MEMBER_VALUE = 1,

1,

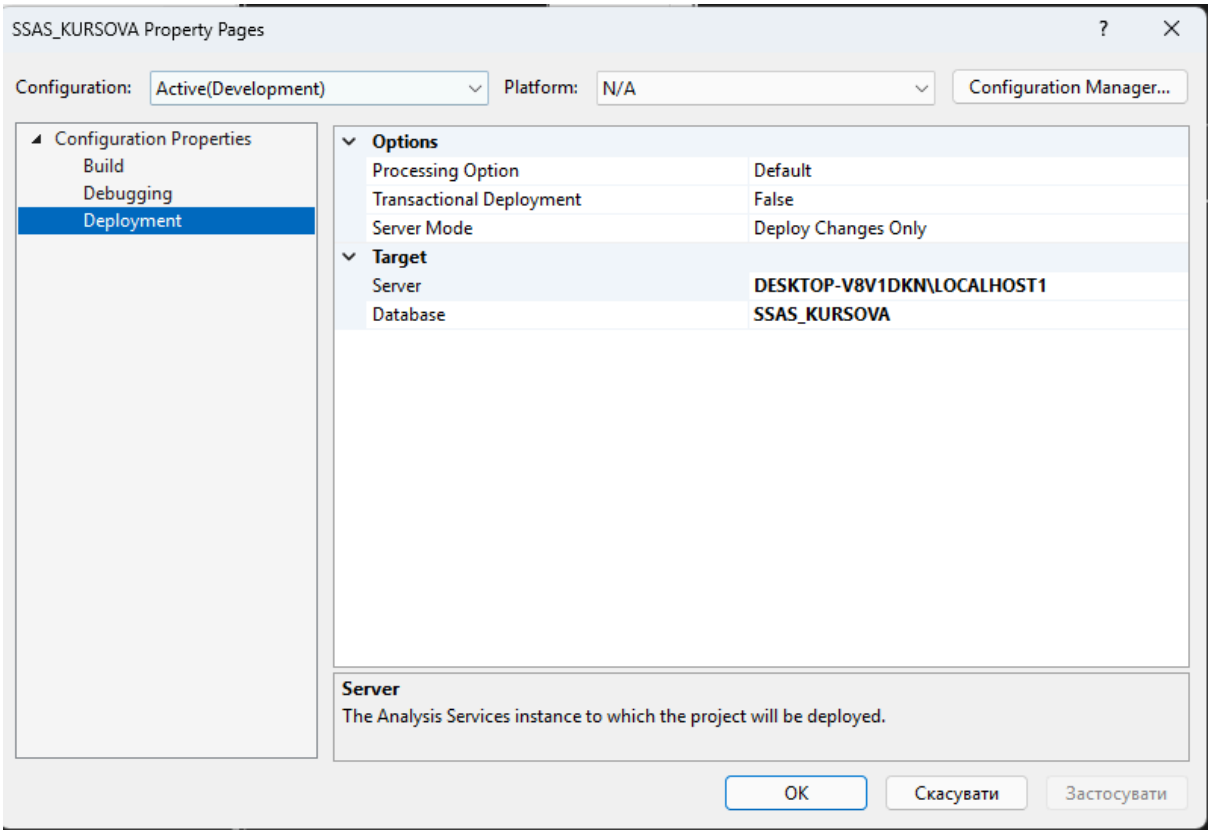
0

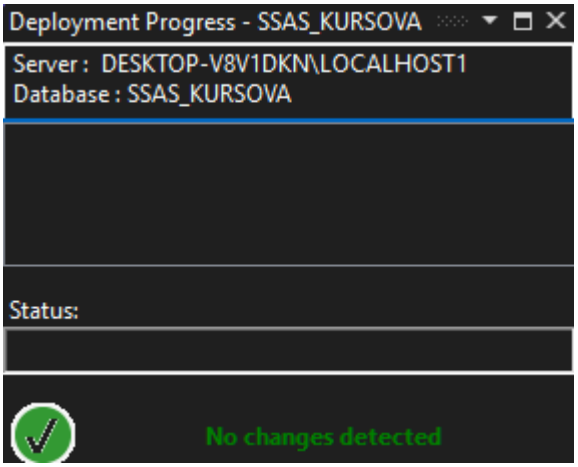
)

[PaidInvoices]=([Measures].[Fact Invoice Count], [Dim Payment Status].[Payment Status SK].&[1])

[UnpaidInvoices]=([Measures].[Fact Invoice Count], [Dim Payment Status].[Payment Status SK].&[2])

Задеплоїмо проєкт. Для цього в properties вкажу шлях деплою:



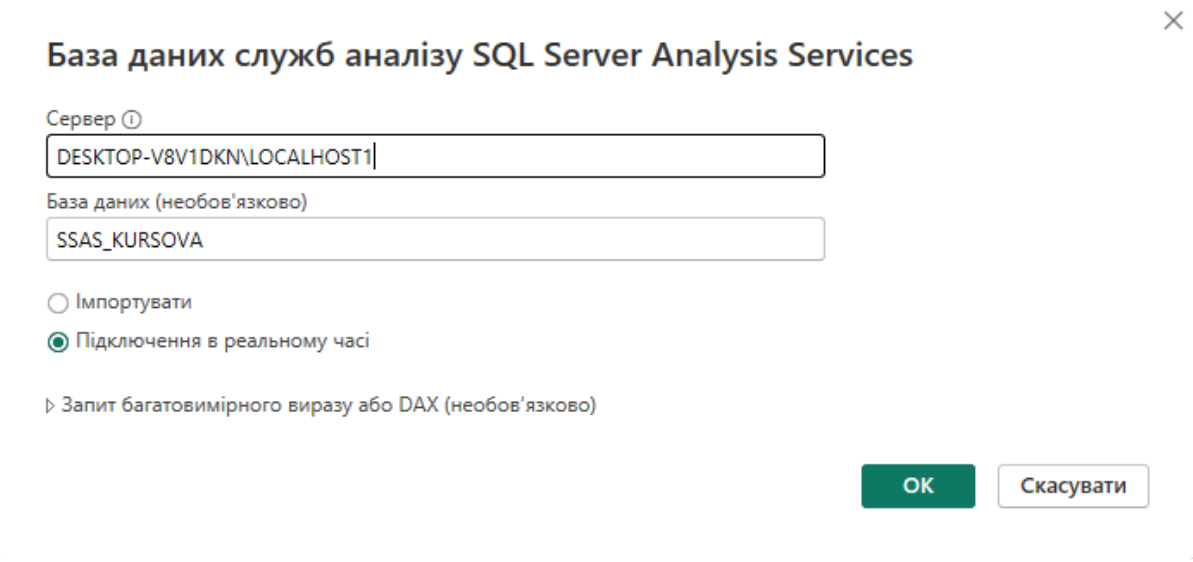


Деплой успішний та без помилок.

Аналітичні звіти

Для побудови звітів спочатку було завантажено та встановлено програму Power BI.

Підключено до OLAP-кубу в режимі підключення в реальному часі:



Було побудовано такі звіти:

1. Список орендарів з інформацією про оренду та борги. Показує повне ім’я орендаря, приміщення, яке він орендував, або орендує та тип. А також наявність не сплаченого боргу за оренду. Інформацію про борг було отримано через формулу різниці суми платежу в рахунку, яку потрібно сплатити, та скільки оплачено орендарем.

Full Name	Room Number	Debt	Type Name
Abel Vaughan	07770	0,00	Conference Room
Abel Wade	58238	0,00	Storage
Abel Walls	50456	0,00	Storage
Abel Ward	63778	0,00	Conference Room
Abel Ware	30744	0,00	Office
Abel Weber	45427	0,00	Warehouse
Abel Welch	18273	0,00	Conference Room
Abel West	88411	60 846,82	Retail
Abel Yates	70182	0,00	Warehouse
Abel Zimmerman	07245	0,00	Conference Room
Abigail Acosta	77503	0,00	Warehouse
Abigail Adams	86198	0,00	Conference Room
Abigail Andrade	15628	9 937,79	Retail
Abigail Avila	07865	0,00	Office
Abigail Avila	77812	0,00	Warehouse
Abigail Banks	38910	0,00	Residential
Abigail Barber	94478	0,00	Conference Room
Abigail Barker	99493	0,00	Warehouse
Abigail Barrett	30328	0,00	Storage
Abigail Bartlett	06348	0,00	Conference Room
Abigail Becker	10490	0,00	Retail
Abigail Berry	08254	0,00	Storage
Abigail Berry	13332	0,00	Residential
Abigail Booker	18456	0,00	Retail
Abigail Boyer	36149	0,00	Retail
Abigail Branch	46980	0,00	Warehouse
Abigail Bray	51016	0,00	Warehouse
Abigail Bridges	00285	0,00	Retail
Abigail Butler	97827	0,00	Warehouse
Abigail Campos	05502	0,00	Retail
Abigail Carrillo	55650	0,00	Conference Room
Abigail Carroll	45585	0,00	Residential
Abigail Case	00418	0,00	Warehouse

2. Стан приміщень на певну дату. Показує за 30 травня 2019 року чи оплачена оренда приміщень.

Full Date	Room Number	Type Name	Status Name
2019-05-30	00055	Storage	Unpaid
2019-05-30	00854	Warehouse	Paid
2019-05-30	02063	Office	Unpaid
2019-05-30	03178	Residential	Paid
2019-05-30	03600	Retail	Unpaid
2019-05-30	04945	Conference Room	Unpaid
2019-05-30	05686	Residential	Paid
2019-05-30	06966	Storage	Paid
2019-05-30	07545	Office	Unpaid
2019-05-30	07849	Conference Room	Unpaid
2019-05-30	09521	Residential	Paid
2019-05-30	10324	Conference Room	Paid
2019-05-30	10756	Office	Unpaid
2019-05-30	11276	Office	Unpaid
2019-05-30	11356	Retail	Unpaid
2019-05-30	12667	Warehouse	Paid
2019-05-30	14776	Warehouse	Unpaid
2019-05-30	15671	Storage	Unpaid
2019-05-30	17981	Office	Paid
2019-05-30	21056	Conference Room	Paid
2019-05-30	21274	Storage	Paid
2019-05-30	21495	Conference Room	Unpaid
2019-05-30	22827	Retail	Paid
2019-05-30	24271	Conference Room	Paid
2019-05-30	24440	Retail	Paid
2019-05-30	33729	Conference Room	Paid
2019-05-30	33849	Storage	Unpaid
2019-05-30	36050	Conference Room	Paid
2019-05-30	37118	Warehouse	Unpaid
2019-05-30	37308	Retail	Paid
2019-05-30	37996	Office	Paid
2019-05-30	38192	Office	Paid
2019-05-30	40789	Warehouse	Paid
2019-05-30	41800	Office	Unpaid

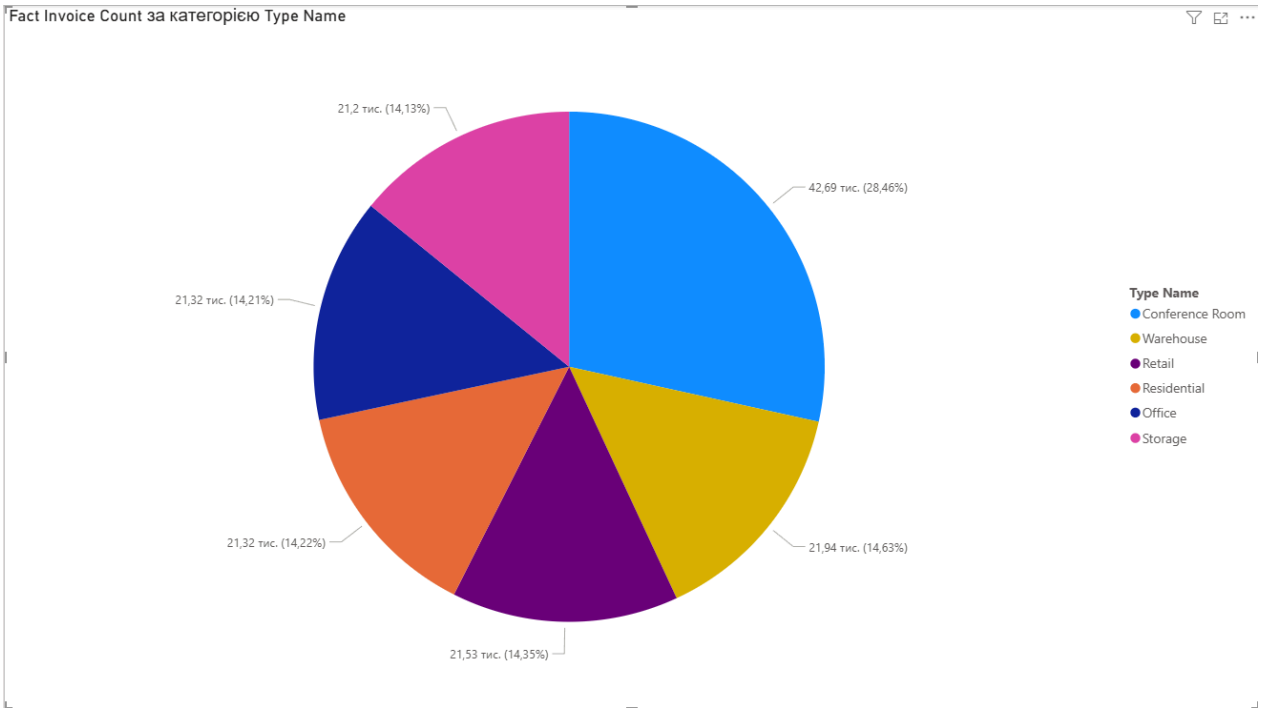
3. Довідка орендаря про історію оренди та платежів. Показує інформацію про орендаря з унікальним номером 4511, повне ім’я, приміщення які він орендував, та їх типи, дати терміну дії договору на ці приміщення. Також

рахунок за оренду.

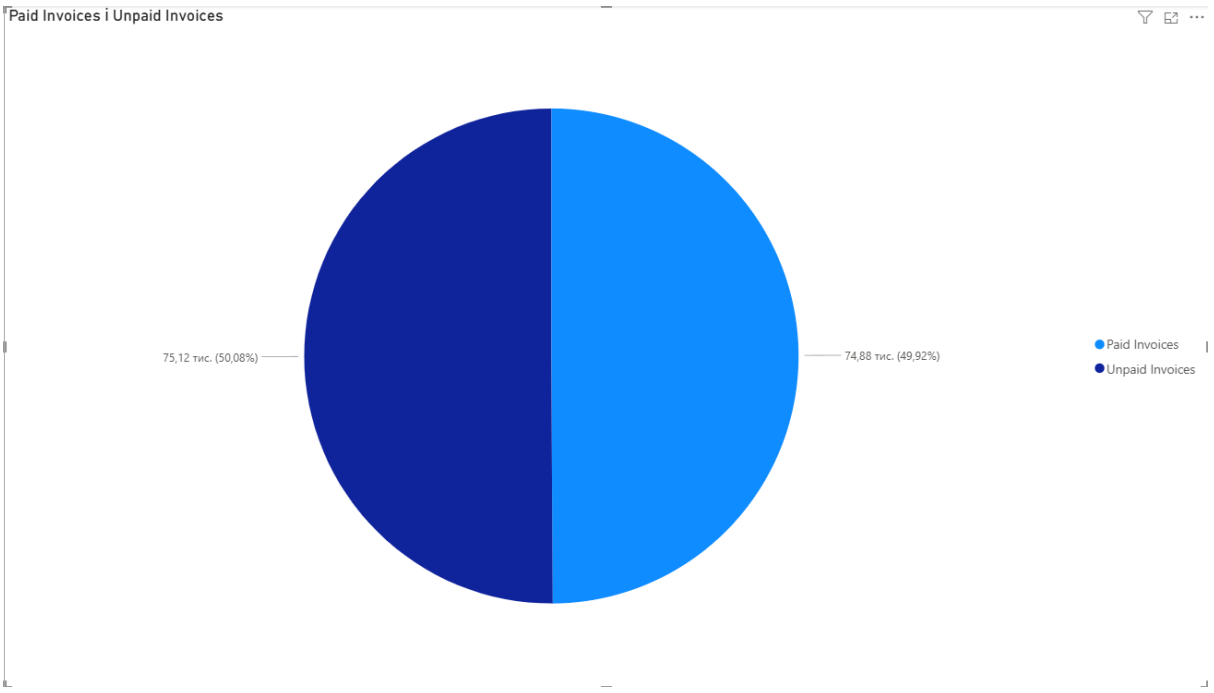
Full Name	Tenant SK	Room Number	Type Name
Diana Ross	4511	58158	Office
Diana Ross	4511	58158	Office

Amount	Start Date	End Date
13509.06	2024-12-10	2025-12-10
86927.54	2023-09-02	2024-09-02

4. Аналіз завантаженості приміщень. Було створено Pie-chart діаграму, яка показує які типи приміщень частіше винаймають в оренду.



5. Фінансовий звіт по орендарях. Було створено Pie-chart діаграму, яка показує відношення скільки приміщень оплачені чи ні на певний момент.


Paid Invoices і Unpaid Invoices
75,12 тис. (50,08%)
74,88 тис. (49,92%)
Paid Invoices
Unpaid Invoices

Висновки

У межах курсової роботи було виконано комплексне дослідження предметної області оренди приміщень та здійснено побудову інформаційно-аналітичної системи для подальшого аналізу даних. Предметна область охоплює процеси обліку приміщень, орендарів, договорів оренди, нарахування орендних платежів, фактичних оплат та контролю фінансового стану орендарів упродовж тривалого періоду часу. Особливу увагу приділено збереженню історичності даних та можливості аналізу інформації в динаміці за кілька років.

На основі вихідної оперативної бази даних Rent, створеної в середовищі SQL Server Management Studio, було спроєктовано сховище даних RentDW з використанням багатовимірного підходу. Архітектура сховища реалізована у вигляді схеми зірки, що забезпечує зручність аналітичних запитів та високу продуктивність при роботі з великими обсягами інформації. У сховищі використано таблиці вимірів, які описують ключові сутності предметної області, та таблиці фактів, що акумулюють числові показники, необхідні для аналізу фінансових та операційних процесів.

Для наповнення сховища даних було створено SSIS-проєкт у середовищі Visual Studio, у якому реалізовано процеси витягання, перетворення та завантаження даних (ETL). У процесі завантаження застосовано механізм Slowly Changing Dimensions, що дозволило коректно зберігати історію змін атрибутів орендарів без втрати актуальних даних. Також використано перетворення Lookup для забезпечення цілісності даних та правильного зв’язування фактів із відповідними вимірами.

На основі сховища даних було створено OLAP-куб із використанням існуючих таблиць фактів та вимірів. Таблиці фактів використано як групи мір, а таблиці вимірів — для формування багатовимірної структури аналізу.

Це дало змогу виконувати аналітичні операції за різними зрізами, зокрема за орендарями, приміщеннями, договорами та часовими періодами. Додаткові аналітичні показники та розрахункові міри були реалізовані за допомогою MDX-формул.

У результаті виконаної роботи створена система дозволяє формувати обов’язкові аналітичні звіти, зокрема аналіз заборгованостей орендарів, оцінку завантаженості приміщень, фінансові підсумки та історію оренди. Отримане рішення може бути використане як основа для підтримки управлінських рішень у сфері оренди нерухомості та подальшого розширення аналітичного функціоналу.

Використані джерела:

1. Інформація про SQL Server Integration Services:

https://uk.wikipedia.org/wiki/SQL_Server_Integration_Services

2 . Інформація про Microsoft Analysis Services:

https://en.wikipedia.org/wiki/Microsoft_Analysis_Services

https://uk.wikipedia.org/wiki/OLAP

3. Про Microsoft SQL Server:

https://uk.wikipedia.org/wiki/Microsoft_SQL_Server

Додатки:

Додаток 1 - sql-скрипти для створення бази даних Rent та таблиць:

```
CREATE DATABASE Rent;
GO

USE Rent;
GO

CREATE TABLE RoomType (
   RoomTypeID INT IDENTITY PRIMARY KEY,
   TypeName NVARCHAR(100) NOT NULL
);

CREATE TABLE Room (
   RoomID INT IDENTITY PRIMARY KEY,
   RoomTypeID INT NOT NULL,
   RoomNumber NVARCHAR(50) NOT NULL,
   Area DECIMAL(8,2) NOT NULL,
   IsActive BIT NOT NULL DEFAULT 1,

   CONSTRAINT FK_Room_RoomType
      FOREIGN KEY (RoomTypeID) REFERENCES RoomType(RoomTypeID)
);

CREATE TABLE Tenant (
   TenantID INT IDENTITY PRIMARY KEY,
   FullName NVARCHAR(200) NOT NULL,
   Phone NVARCHAR(50),
   Email NVARCHAR(100)
```

```
);


CREATE TABLE LeaseContract (
    ContractID INT IDENTITY PRIMARY KEY,
    TenantID INT NOT NULL,
    RoomID INT NOT NULL,
    StartDate DATE NOT NULL,
    EndDate DATE NULL,
    RentPrice DECIMAL(10,2) NOT NULL,

    CONSTRAINT FK_Contract_Tenant
        FOREIGN KEY (TenantID) REFERENCES Tenant(TenantID),

    CONSTRAINT FK_Contract_Room
        FOREIGN KEY (RoomID) REFERENCES Room(RoomID)
);


CREATE TABLE Invoice (
    InvoiceID INT IDENTITY PRIMARY KEY,
    ContractID INT NOT NULL,
    InvoiceDate DATE NOT NULL,
    Amount DECIMAL(10,2) NOT NULL,
    IsPaid BIT NOT NULL DEFAULT 0,

    CONSTRAINT FK_Invoice_Contract
        FOREIGN KEY (ContractID) REFERENCES LeaseContract(ContractID)
);
```

```
CREATE TABLE Payment (
    PaymentID INT IDENTITY PRIMARY KEY,
    InvoiceID INT NOT NULL,
    PaymentDate DATE NOT NULL,
    Amount DECIMAL(10,2) NOT NULL,

    CONSTRAINT FK_Payment_Invoice
        FOREIGN KEY (InvoiceID) REFERENCES Invoice(InvoiceID)
);
```

Додаток 2 - sql-скрипт для корекції даних:

```
UPDATE LeaseContract
SET EndDate = DATEADD(YEAR, 1, StartDate)
WHERE EndDate IS NOT NULL
  AND StartDate >= EndDate;
```

Додаток 3 - sql-скрипти для створення бази даних Rent та таблиць:

```
CREATE DATABASE RentDW;
GO

USE RentDW;
GO



CREATE TABLE DimTenant (
    TenantSK INT IDENTITY PRIMARY KEY,
```

```
    TenantBK INT NOT NULL,

    FullName NVARCHAR(200),
    Phone NVARCHAR(50),
    Email NVARCHAR(100),

    StartDate DATE NOT NULL,
    EndDate DATE NULL,
    IsCurrent BIT NOT NULL
);

CREATE TABLE DimRoom (
    RoomSK INT IDENTITY PRIMARY KEY,
    RoomBK INT NOT NULL,

    RoomNumber NVARCHAR(50),
    Area DECIMAL(8,2)
);


CREATE TABLE DimRoomType (
    RoomTypeSK INT IDENTITY PRIMARY KEY,
    RoomTypeBK INT NOT NULL,

    TypeName NVARCHAR(100)
);


CREATE TABLE DimDate (
```

```
    DateSK INT PRIMARY KEY,
    FullDate DATE NOT NULL,
    Year INT,
    Month INT,
    Day INT
);


CREATE TABLE DimContract (
    ContractSK INT IDENTITY PRIMARY KEY,
    ContractBK INT NOT NULL,

    StartDate DATE,
    EndDate DATE,
    MonthlyRent DECIMAL(10,2)
);


CREATE TABLE DimPaymentStatus (
    PaymentStatusSK INT IDENTITY PRIMARY KEY,
    StatusName NVARCHAR(50)
);


CREATE TABLE FactInvoice (
    InvoiceSK INT IDENTITY PRIMARY KEY,

    DateSK INT NOT NULL,
    TenantSK INT NOT NULL,
```

```
    RoomSK INT NOT NULL,
    ContractSK INT NOT NULL,
    PaymentStatusSK INT NOT NULL,

    Amount DECIMAL(10,2),

    CONSTRAINT FK_FactInvoice_Date
        FOREIGN KEY (DateSK) REFERENCES DimDate(DateSK),

    CONSTRAINT FK_FactInvoice_Tenant
        FOREIGN KEY (TenantSK) REFERENCES DimTenant(TenantSK),

    CONSTRAINT FK_FactInvoice_Room
        FOREIGN KEY (RoomSK) REFERENCES DimRoom(RoomSK),

    CONSTRAINT FK_FactInvoice_Contract
        FOREIGN KEY (ContractSK) REFERENCES DimContract(ContractSK),

    CONSTRAINT FK_FactInvoice_Status
        FOREIGN KEY (PaymentStatusSK) REFERENCES
DimPaymentStatus(PaymentStatusSK)
);



CREATE TABLE FactPayment (
    PaymentSK INT IDENTITY PRIMARY KEY,

    DateSK INT NOT NULL,
```

```
    TenantSK INT NOT NULL,
    InvoiceSK INT NOT NULL,

    Amount DECIMAL(10,2),

    CONSTRAINT FK_FactPayment_Date
        FOREIGN KEY (DateSK) REFERENCES DimDate(DateSK),

    CONSTRAINT FK_FactPayment_Tenant
        FOREIGN KEY (TenantSK) REFERENCES DimTenant(TenantSK),

    CONSTRAINT FK_FactPayment_Invoice
        FOREIGN KEY (InvoiceSK) REFERENCES FactInvoice(InvoiceSK)
);



DECLARE @StartDate DATE = '2019-01-01';
DECLARE @EndDate   DATE = '2024-12-31';

WHILE @StartDate <= @EndDate
BEGIN
    INSERT INTO DimDate (DateSK, FullDate, Year, Month, Day)
    VALUES (
        YEAR(@StartDate) * 10000
        + MONTH(@StartDate) * 100
        + DAY(@StartDate),
        @StartDate,
        YEAR(@StartDate),
        MONTH(@StartDate),
```

```
        DAY(@StartDate)
    );

    SET @StartDate = DATEADD(DAY, 1, @StartDate);
END;
```