Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп'ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Модель OLAP-гіперкуба системи аналізу банківських операцій

(тема)

Виконав:

студент II курсу, групи СПм-21-1

Лифар Д.С.
(прізвище, ініціали)

Спеціальність

123 «Комп'ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Мартовицький В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Лифарю Дмитру Сергійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Модель OLAP-гіперкуба системи аналізу банківських операцій

затверджена наказом по університету від “ 07 ” листопада 2022 р. № 1454 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2022 р.

3. Вхідні дані до роботи 1) OLAP-технології; 2) база даних; 3) ETL-процес.

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз сучасних систем аналітики;
2) аналіз предметної області;
3) розробка ETL-процесу;
4) розробка OLAP-моделі;
5) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та існуючих рішень	08.11.22-11.11.22	
2	Вибір технологій та інструментів	12.11.22-13.11.22	
3	Розробка ETL-процесу	14.11.22-22.11.22	
4	Розробка OLAP-моделі	23.11.22-30.11.22	
5	Проведення експериментів з тестовими даними	31.11.22-02.12.22	
6	Оформлення матеріалів кваліфікаційної роботи	03.12.22-06.12.22	
7	Подання кваліфікаційної роботи керівникові	07.12.22-08.12.22	
	та її попередній захист		
8	Подання кваліфікаційної роботи на	09.12.22-12.12.22	
	рецензування		

Дата видачі завдання 07 листопада 2022 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Мартовицький В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 99 с., 36 рис., 1 дод., 22 джерела.

БАЗА ДАНИХ, OLAP, КУБ, ВИМІР, МЕТРИКА, КАЛЬКУЛЯЦІЯ, АТРИБУТ, ПРОЦЕДУРА, ETL, СЕРВЕР, АРХІТЕКТУРА, BUSINESS INTELLIGENCE, SSMS, SSAS, SQL, ТРАНЗАКЦІЯ, БАНК.

Метою кваліфікаційної роботи є розробка моделі OLAP-компоненту та бази даних, що виступає у ролі обробника, підготовки та трансформації отриманих вхідних даних банківських операцій, а також є основою і джерелом для подальшої побудови на ній моделі для аналітичних цілей за допомогою можливостей Business Intelligence технологій.

У ході виконання кваліфікаційної роботи було розроблено OLAP-компонент та базу даних. Розроблено структуру, функції та процедури бази даних, які обробляють вхідну інформацію до належної форми для її подальшої обробки у OLAP-частині моделі. Призначенням даної моделі є надання аналітичної форми отриманих даних банківських операцій, яка у подальшому може бути використана у клієнтських додатках.

ABSTRACT

Master’s thesis: 99 pages, 36 figures, 1 appendice, 22 sources.

DATABASE, OLAP, CUBE, DIMENSION, METRIC, CALCULATION, ATTRIBUTE, PROCEDURE, ETL, SERVER, ARCHITECTURE, BUSINESS INTELLIGENCE, SSMS, SSAS, SQL, TRANSACTION, BANK.

The major goal of this thesis is to develop a model of the OLAP-component and a database that acts as a processor, prepare and transforms the received input data of banking operations, and what is the basis and source for further building a model on it for analytical purposes using the capabilities of Business Intelligence technologies also.

In the course of the qualification work, an OLAP-component and a database were developed. Have been developed the structure, functions and procedures of the database, which processes the input information into the appropriate form for its further processing in the OLAP-part of the model. The purpose of this model is to provide an analytical form of the received data of banking operations, which can be used in client applications in the future.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Вимір – вісь куба, що виражається унікальними значеннями конкретних атрибутів

Калькуляція – агрегація значень, визначена за допомогою мови MDX, оперуючи об’єктами куба

Куб – об’єкт, сутність, що представляє собою сукупність вимірів, метрик, калькуляцій, властивостей, джерел у контексті OLAP

Метрика – агрегація значень, що основана на значеннях з бази даних

СУБД – система управління базою даних

BI – узагальнене позначення методів, інструментів та засобів, що забезпечують обробку великих об’ємів транзакційної інформації (англ. Business Intelligence)

ETL – один з основних процесів в керуванні сховищами даних (англ. Extract Transform Load)

MDX – мова запитів до багатомірних структур даних (англ. MultiDimensional eXpressions)

OLAP – технології обробки інформації для представлення аналітичних даних у формі багатовимірного об’єкту (англ. Online Analytical Processing)

SSAS – служби, набір засобів для роботи з OLAP і Data Mining технологіями (англ. SQL Server Analysis Services)

SSMS – інтегроване середовище для адміністрування SQL інфраструктури (англ. SQL Server Management Studio)

SQL – мова запитів для роботи з базами даних (англ. Structured Query Language)

ВСТУП

На сьогоднішній день у світі існує безліч способів та методів аналізу даних для будь-якої цілі або мети. Хтось проводить експерименти у галузі біології задля визначення закономірностей поведінки тих чи інших організмів, хтось досліджує тенденцію цін на конкретні товари, їх залежність від різних факторів – починаючи від пори року і закінчуючи логістичними витратами на паливо та обслуговування транспортних засобів, а хтось на основі аналітичних даних може давати прогнози щодо ринкової ціни криптовалют – від популярних та найвідоміших, до найменших та найновіших, що тільки нещодавно почали своє існування.

Всі ці задачі мають своє призначення (підвищення ефективності, отримання будь-якої вигоди, підтвердження достовірності теорій, перевірка чинності, прогнозування та багато чого іншого) і будуються на основі інформації різноманітного розміру – від кількох сотень експериментів до мільйонів та мільярдів записів транзакцій, статистики, щоденних чи погодинних даних тощо.

Може здатися, що для вирішення названих вище цілей можна створити один єдиний інструмент, який задовольнить потреби кожного у сфері аналізу та обробки даних, проте, нажаль, це не є можливим – будь-яка сфера діяльності людини є унікальною не тільки з фізичного чи логічного боку, а й зі сторони того, які саме дані, в якій кількості, з якою частотою вони повинні оброблятися та надавати вихідну інформацію тощо.

Розглянемо в якості прикладу Microsoft Excel – усім відому офісну програму для роботи з таблицями. Вона надає широкий спектр інструментів для офісної роботи, кількість можливостей та функцій вражає, підходить для багатьох звичних у повсякденному житті задач, проте у будь-якого продукту є свої ліміти та обмеження. Наприклад, на одній робочій сторінці максимальна кількість рядків дорівнює 1048576, а кількість стовпців – 16384.

Для великих обсягів даних одного мільйону рядків буде недостатньо, а розділення даних на окремі сторінки може не стати вирішенням проблеми. Виконання спільних дій чи обчислень, які будуть повинні охоплювати дані на кількох сторінках будуть або неможливими, або виконані за допомогою проміжних сторінок та значень. Наприклад, якщо це вирахування суми на одному стовпцю, то спочатку потрібно обчислити суму на кожній сторінці, а потім вже скласти всі ці проміжні значення сум в одне ціле. Ці незручності виникнуть тому що сторінки будуть існувати як незалежні дані (будуть об’єднані лише спільною робочою книгою), тобто знаходяться в одному Excel файлі. На заміну цього та для вирішення подібних проблем і незручностей для роботи та розрахунків на основі великого обсягу даних стають спеціалізовані сховища даних.

Також існує інша проблема, що може впливати на ефективність роботи – це можливості продуктивності апаратного забезпечення пристрою. Наприклад, при повному заповненні по кількості рядків, у разі накладання будь-якого фільтру, зміни стилю для обраного стовпця, або ж умовного форматування по умові, або ще якихось дій над всім об’ємом даних – завантаженість та тимчасове зависання програми неминуче, адже пристрою потрібно обробити кожен рядок, провести вказану операцію над кожним з них у режимі реального часу та повернути результат.

Навіть якщо можна вирішити питання ліміту рядків, або ж потрібні дані мають меншу їх кількість, яка у результаті зможе розміститися на одній робочій сторінці, а також вирішиться проблема питання з продуктивністю системи то далі, в залежності від сфери або галузі не всі задачі в аналітичному напрямку зможуть бути вирішені.

Саме тому створення аналітичної системи [1] завжди буде актуальним рішенням, тому що для кожної сфери, що потребує обробки величезного обсягу інформації [2], що надається, потрібні індивідуальні налаштування в залежності від багатьох факторів – починаючи від країни замовника, сфери діяльності, об’єму вхідних даних, частоти отримання цих даних, частоти

потрібного оновлення, критичності оновлення у конкретний час, бізнес цілями, закінчуючи власними побажаннями та особливостями роботи замовника.

Така система може складатися з множини компонентів та додатків, або ж це буде як десктопний додаток, або ж веб-сайт у його кінцевому вигляді. Проте основна обчислювальна потужність такої системи для зберігання, підготовки та обробки вхідної інформації у будь-якому випадку буде знаходитись у базі даних, як у сховищі даних, а також у аналітичній OLAP-системі, яка буде зберігати:

- повний набір вимірів;
- їх ієрархії у випадку присутності таких;
- метрики та калькуляції.

Також, при всьому цьому, OLAP ще заздалегідь буде вираховувати та зберігати агрегації на всіх її можливих точках – точках перетину кожного с кожним вимірів.

Метою кваліфікаційної роботи є розробка моделі OLAP-компоненту та бази даних, що виступає у ролі обробника, підготовки та трансформації отриманих вхідних даних банківських операцій, а також є основою і джерелом для подальшої побудови на ній моделі для аналітичних цілей за допомогою можливостей Business Intelligence технологій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність аналітичних даних для банківської справи

Точне та своєчасне прийняття рішень у банківській сфері означає раціоналізацію операцій, зниження ризиків, підвищення прибутку та покращення якості обслуговування клієнтів.

Щоб директори та керівники могли приймати більш оптимальні та конструктивні рішення, їм потрібне відповідне джерело інформації, що може допомогти у цьому – надати детальні, своєчасні, лише необхідні дані у розрізі тієї чи іншої проблеми. Саме для таких рішень у нагоді стає бізнес-аналітика, що також з технічного боку має назву Business Intelligence.

Business Intelligence (BI) – це узагальнене позначення методів, інструментів та засобів, що забезпечують обробку великих об’ємів інформації, яка створюється для бізнес цілей з метою розвитку тієї чи іншої сфери, покращення положення, примноження прибутку, зменшення кумулятивних витрат.

Інвестуючи у правильну BI-технологію, банки можуть отримати аналіз та інформацію, які необхідні для адаптації до ринку, збільшення задоволеності потреб клієнтів та розвитку бізнесу.

Business Intelligence відноситься до інструментів та технологій, що використовуються для більш глибокого сприйняття та розуміння діяльності компанії з операційного боку для того, щоб допомогти таким чином приймати найкращі та найбільш раціональніші рішення зі стратегічної точки зору. Бізнес-аналітика виходить за звичні рамки лише цифр та аналітики. Вона може представляти такі методи, що у наслідку можуть надавати конструктивні, раціональні висновки з урахуванням аналізу даних, які можуть легко зрозуміти (і швидко прийняти правильне рішення) керівники компанії – без необхідності глибоких знань у статистиці чи математиці.

Через величезний обсяг даних, таких як інформація про клієнтів та

транзакції, що створюються і оперуються у банківській сфері, керуючим особам та установам непрактично та нераціонально аналізувати такі об’єми інформації власноруч. В теперішній час технології дозволяють отримувати аналітичну інформацію про ті чи інші дані та автоматизувати і вирішувати багато бізнес-аналітичних процесів. BI-рішення зараз можуть запропонувати ефективні та гнучкі способи управління своїми маркетинговими кампаніями, активами та ризиками тощо.

Проте бізнес-аналітика [3] – це не лише звіти та статистика. Актуальні Business Intelligence системи включають дешборди, які надають актуальну статистику, дозволяють аналізувати продуктивність у будь-який момент часу, для того, щоб мати можливість негайно приймати певні рішення, які можуть суттєво вплинути на майбутні витрати або прибутки. Нижче наведені найбільші з основних переваг використання дешбордів для бізнес-аналітики:

- моніторинг відділень – можна завжди «тримати руку на пульсі», тому що маючи оперативні та оновлені дані з усіх відділень банку, можливо виконувати безліч задач, що потребують управління та контролю, наприклад, отримати середній час очікування в черзі та інші відомості, що можуть відстежуватися, бути наданими та впливати на якість обслуговування клієнтів;

- використання робочих місць – у разі наявності доступу до даних з пристроїв співробітників, мається можливість оцінювати якість роботи, яку вони виконують;

- моніторинг продуктивності – оцінювати ситуацію та порівнювати продуктивності між відділеннями, філіями та співробітниками тощо;

- відстеження бронювання – перегляд останніх зареєстрованих бронювань, бронювань візиту, тощо;

- поточний стан – перегляд аналітичних даних з різних джерел в одному єдиному місці з використанням фільтрування починаючи з міста, регіону, відділення та закінчуючи навіть категорією транзакції (наприклад, виплати по кредиту, закупівлі, персональний та бізнес-банкінги, обмін

валюти та кредити). За допомогою статистики є можливість дізнатися середній час очікування, скільки клієнтів очікує відповідей або тих чи інших дій від співробітників, ефективність роботи служби підтримки, коефіцієнти продуктивності у будь-якому розрізі даних і багато іншого.

Такі потужні інструменти як онлайн дешборди [4] разом із розширеними звітами, аналітикою та статистикою надають керівникам фінансових установ зручні способи контролю, моніторингу та керування ресурсами для покращення якості обслуговування клієнтів, збільшення прибутку, зменшення витрат тощо.

Бізнес-аналітика є вигідною з економічної та оптимізаційної точки зору, що, як наслідок, приносить величезні плюси та переваги банкам і подібним установам, а саме:

- вимірювана рентабельність інвестицій – середня 5-річна рентабельність інвестицій за допомогою бізнес-аналітики може становити 112%;

- залучення нових клієнтів – компанії, що керують та аналізують власні дані за допомогою статистики та аналітики, в 24 рази частіше поповнюються новими клієнтами, що у банківській сфері з великою кількістю конкурентів є дуже важливим фактором прибутку та іміджу;

- підвищення конкурентоспроможності – банки, які не використовують рішення Business Intelligence, ризикують залишитися без суттєвих змін у економічній та перспективній складовій;

- управління ризиками – деякі бізнес-інструменти використовуються установами та їх мережами для аналізу ризиків щодо кредитування фізичних чи юридичних осіб, іншим це може бути у нагоді для відстеження та контролю фінансового ринку. Це означає, що особи або працівники, які ухвалюють рішення у видачі кредиту, можуть використовувати ці інструменти та звіти для оцінки та зниження ризику;

- розуміння цифр – зміна представлення даних із нудних, однотипних цифр, що зливаються в один суцільний текст на візуальні звіти, у яких легко

розібратися та просто зрозуміти, а також за потреби перетворити на стратегічні, оптимізаційні дії та плани, які у майбутньому допоможуть досягти поставлених цілей;

- більш гнучкі можливості – маючи під рукою моніторинг, що може збирати різноманітну інформацію з системи управління у банку, користувач зможе швидко приймати своєчасні рішення. В результаті можливо скоригувати штатний розклад, скоротити час очікування, збільшити ефективність та якість обслуговування, підвищити клієнтоорієнтованість, зменшити витрати та збільшити кількість продаж;

- персоналізація клієнтської роботи – у разі наявності аналітики, співробітники можуть особисто знаходити підхід до кожного клієнта, краще розуміти його можливості, проблеми та потреби, пропонувати та здійснювати більш раціональні продажі відповідних продуктів на основі відповідного профілю та історії;

- за допомогою аналітичних даних Business Intelligence, стає можливим здійснити перехід від фізичного відділення до цифрового, і, як наслідок, почати пропонувати збільшений спектр послуг для будь-яких потреб клієнта у новому та сучаснішому форматі;

- ефективний ресурсний розподіл – складання загальної картини щодо активності та продуктивності транзакцій залежно від будь-яких критеріїв у будь-якому розрізі даних: часу, місця розташування, сегмента тощо. Отримуючи інформацію про робоче навантаження, послуги, доступність, можна направити ресурси в те місце, де вони необхідні, а також належним чином забезпечувати та покращувати обслуговування клієнтів, одночасно збільшуючи прибуток;

- завдяки більш глибокому розумінню споживачів, надавати їм ті види обслуговування, які для них є більш задовольняючими та раціональними, що також покращить можливість і зменшить складність у рекомендаціях актуальні продуктів. В результаті може збільшитися кількість продажів – неважливо або це перший продаж, або постійний клієнт;

- вимірювання задоволеності клієнтів – якщо інтегрувати відгуки та побажання, то потім можна дізнатися, як досвід, який ними був отриманий, може впливати на успіхи компанії, а також знайти ті місця та прогалини, де необхідно сфокусуватися на покращеннях щодо надання послуг.

Проте, звичайно, не всі платформи Business Intelligence створені однаково. Конкретні переваги будуть залежати від того, які функції будуть надаватися з BI-інструментами та як їх використовувати. Маючи просту для огляду та розуміння аналітики, можна аналізувати будь-який критерій, починаючи від регіону, філії, розташування, послуги, типу транзакцій і закінчуючи навіть окремими співробітниками, щоб знати, на яких сферах діяльності та оптимізації праці потрібно зосередитись. Можна виміряти ефективність, знайти сильні та слабкі місця і, як наслідок, вирішити, куди потрібно направити ресурси для максимального підвищення ефективності. Крім того, можливо збільшити кількість продажів, визначити тенденції, покращити якість обслуговування та розширити бізнес.

1.2 Аналіз існуючих систем аналітики

1.2.1 Система аналітики QlikView від BI Consult

QlikView – це BI-платформа з асоціативним пошуком в оперативній пам'яті з вбудованими засобами ETL (Extract-Transform-Load). Платформа Business Discovery призначена для самостійного проведення бізнес-аналізу для всіх корпоративних користувачів. За допомогою програми QlikView можна аналізувати дані та використовувати отримані результати для підтримки рішень (рисунок 1.1).

Можливості рішення QlikView:

- досліджувати тренди показників банківських ризиків;

- забезпечувати автоматичний контроль лімітів, що обмежують рівень банківських ризиків;

- оперативно формувати обґрунтовані відповіді як на стандартні, так і

на нестандартні запити керівництва банку;

- отримувати кількісні оцінки ймовірних втрат (збитків) та/або неотримання запланованих доходів, внаслідок виникнення різних подій, пов'язаних із внутрішніми та/або зовнішніми факторами діяльності банку;

- розраховувати ключові індикатори ризиків, які використовуються спостереження за основними факторами ризику;

- будувати ризик-профіль банку;

- встановлювати рівень толерантності до ризику;

- аналізувати клієнтів у розрізі платіжних систем, за сумами транзакцій, динамікою використання карток, динамікою відкриття карток тощо;

- ранжувати транзакції з прибутковості.

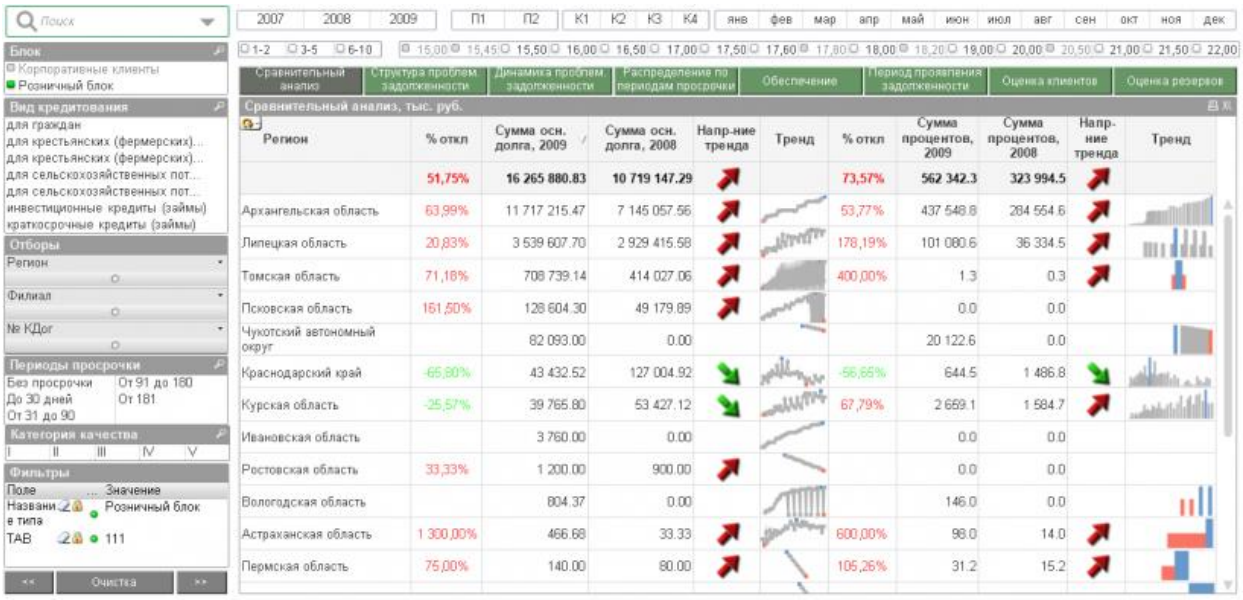


Рисунок 1.1 –Інтерфейс QlikView від BI Consult

Плюси QlikView:

- має аналітичну та динамічну екосистему Business Intelligence;

- дозволяє аналізувати, інтерпретувати та обмінюватися складними даними;

- використовує технологію in-memory, яка пришвидшує зчитування та візуалізацію даних.

Мінуси QlikView:

- використання обчислювальних витрат дорожче, оскільки вимагає більшого простору оперативної пам’яті;

- аналіз даних у реальному часі іноді неможливий за допомогою QlikView;

- відсутня технологія перетягування об’єктів у інтерфейсі, тому робота з ним повільніша;

- вимагає багато додаткових функцій, які потрібно придбати за додаткову плату. Це робить вартість використання QlikView вищою, ніж Tableau;

- має менш дружній інтерфейс користувача із навантаженою інформаційною панеллю;

- користувачі відчувають додаткові труднощі під час розробки програм, вбудовування або інтеграції з іншим програмним забезпеченням;

- має погану та незадовільну онлайн-підтримку.

QlikView більше не є таким прозорим щодо своїх цін, як це було раніше. Ціни, які раніше були сміливо зазначені на веб-сайті, більше не існують. Це коштуватиме організації близько 15500 доларів США для кількох користувачів та 1395 доларів США за одного користувача. Організація також повинна буде заплатити від 8675 до 72000 доларів США за одну або більше серверних ліцензій. Інші послуги та необхідні функції також доступні за додаткову плату.

1.2.2 Система аналітики RoiStat

Roistat – це один із перших гравців на ринку сервісів аналітики. Ця система вважається найбільшою та функціональною завдяки великій кількості можливостей: зведений звіт, проста та мультиканальна аналітика, когортний аналіз, автоматичне управління ставками, інтеграція витрат, спліт-тестування, коллтрекінг, email-трекінг (рисунок 1.2). Якщо потрібний додатковий функціонал, можна замовити платне доопрацювання.

Також це максимально деталізована наскрізна аналітика.

Налаштування сервісу може бути складним для новачків – програма збирає близько 30 показників ефективності. На допомогу клієнтам існують відеоуроки, вебінари, детальна документація щодо налаштування системи.

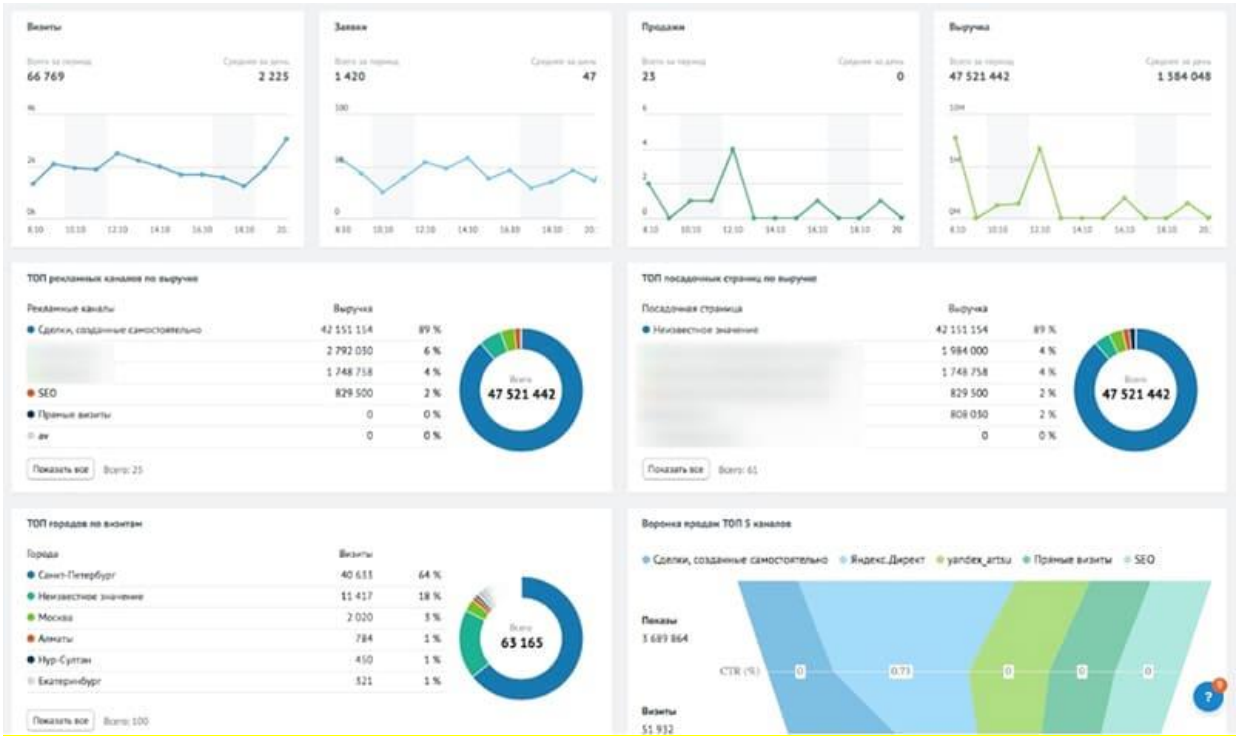


Рисунок 1.2 – Інтерфейс RoiStat

Переваги:

- готова аналітика, можна швидко і не дорого її реалізувати (порівняно з іншими рішеннями);

- дуже багато готових інтеграцій;

- зручні звіти, реально допомагають підвищувати ефективність маркетингу. Зведений звіт надає можливість ознайомитися з користю кожного джерела у взаємодії з клієнтом;

- інтегрується з усіма відомими системами;

- можливість замовити доопрацювання під завдання конкретного бізнесу;

- можливість проводити мультиканальну аналітику – відстежувати, звідки найчастіше приходять відвідувачі та визначати, які з каналів надають найбільший прибуток.

Недоліки:

- не підійде для дуже великого бізнесу, у цьому випадку потрібна налаштована інтеграція через BI;

- велику вага сервісу;

- висока вартість порівняно з іншими продуктами. Є безкоштовний доступ на 14 днів, але потім потрібно обрати тариф. Стартова ціна від 114$ на місяць;

- досить складна інтеграція, у багатьох виникають проблеми;

- є скарги на роботу підтримки.

1.2.3 Система аналітики RoiStat

Alytics – це сервіс наскрізної аналітики (рисунок 1.3), який включає 3 основні модулі:

- наскрізна аналітика: відображає показники від витрат до прибутку в одному звіті за більш ніж 20 джерелами: трафік, дзвінки, продаж CRM, цілі та кошики, залучення відвідувачів, спеціальні показники. Аналізує дані аж до ключової фрази та місця розміщення реклами. Інформація відображається у вигляді графіків, діаграм;

- автоматизація контексту: двостороння інтеграція з Google AdWords, аналіз контекстної реклами за ключовими фразами, 7 способів керування ставками, контроль битих посилань, автоматизація UTM-розмітки;

- колл-трекінг: відображає статистику кожного дзвінка, показує, які джерела наводять найбільше дзвінків, дозволяє відзначати дзвінки тегами (ярликами), записує дзвінки.

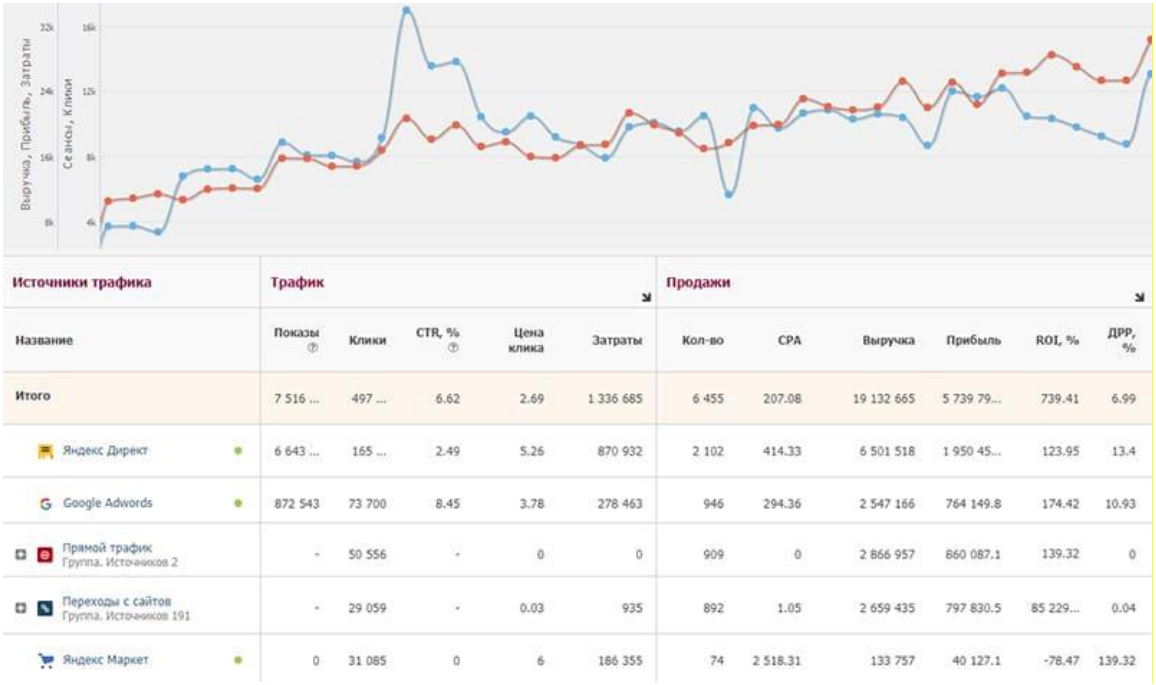
Переваги Alytics:

- крім наскрізної аналітики, є ще система автоматизації контекстної реклами;

- є передача даних у Alytics через postback;

- інтеграція з усіма основними сервісами;

- мінімальна вартість на місяць становить 80$.



Источники трафика	Трафик					Продажи					
Название	Показы	Клики	CTR, %	Цена клика	Затраты	Кол-во	CPA	Выручка	Прибыль	ROI, %	ДРР, %
Итого	7 516 ...	497 ...	6.62	2.69	1 336 685	6 455	207.08	19 132 665	5 739 79...	739.41	6.99
Яндекс Директ	6 643 ...	165 ...	2.49	5.26	870 932	2 102	414.33	6 501 518	1 950 45...	123.95	13.4
Google Adwords	872 543	73 700	8.45	3.78	278 463	946	294.36	2 547 166	764 149.8	174.42	10.93
Прямой трафик Группа. Источников 2	-	50 556	-	0	0	909	0	2 866 957	860 087.1	139.32	0
Переходы с сайтов Группа. Источников 191	-	29 059	-	0.03	935	892	1.05	2 659 435	797 830.5	85 229...	0.04
Яндекс Маркет	0	31 085	0	6	186 355	74	2 518.31	133 757	40 127.1	-78.47	139.32

Рисунок 1.3 – Інтерфейс Alytics

Недоліки Alytics:

- немає стандартної інтеграції з Яндекс Метрикою;
- постійно змінюються умови тарифікації;
- не легко розібратися із сервісом;
- повільна підтримка;
- не найзручніший інтерфейс;
- підійде тільки тим, хто вже має CRM з інтеграцією.

Збір даних походить з CRM, отже, мають бути налаштовані всі інтеграції. Серед функцій присутня деталізована мультиканальна аналітика, можливість надавати інформацію у різних видах, звітність щотижнева та зрозуміла, email-трекінг, керування ставками.

1.2.4 Система аналітики PrimeGate

PrimeGate – це молодий сервіс аналітики, що об'єднує кілька модулів: передовий колтрекінг, аналіз відвідувачів, багатоканальну атрибуцію, керування ставками контекстної реклами, відстеження ефективності офлайн-кампаній, онлайн-чат.

Сервіс PrimeGate має приємний та зручний BI-інтерфейс з наочними звітами та додатковими віджетами (рисунок 1.4). Щоб заощадити час, можна замовити послугу інтеграції сервісу до компанії та навчання співробітників.

Переваги:

- є тестовий період на 7 днів;
- проводять безкоштовну Skype-презентацію сервісу;
- низькі ціни: мінімальна вартість на місяць – 50$.

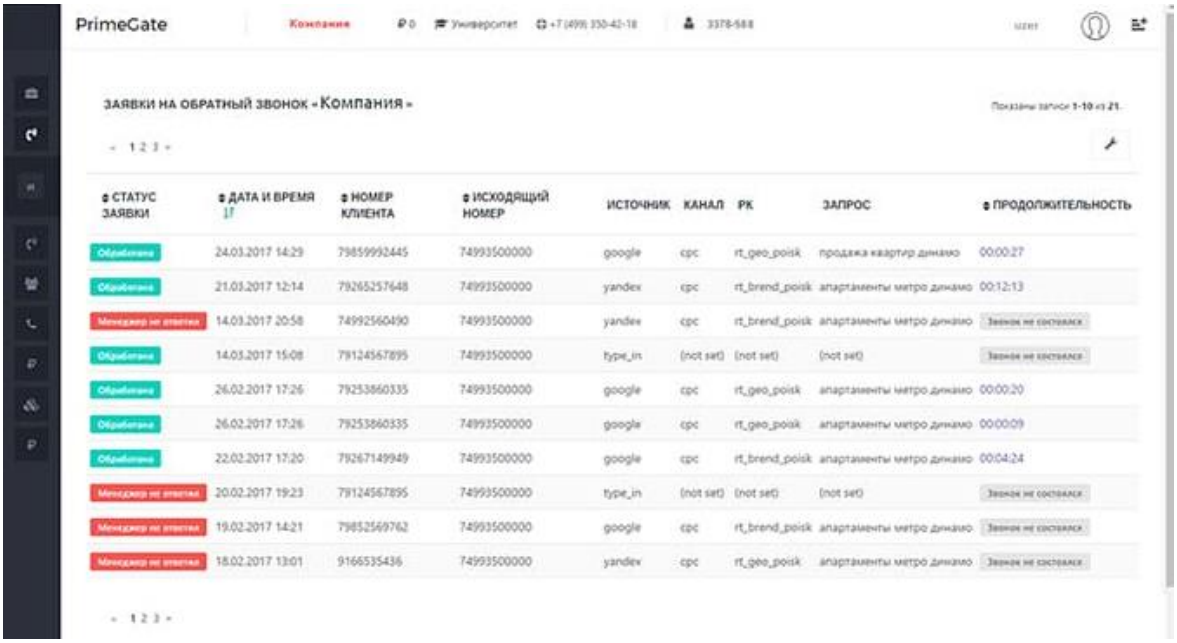


Рисунок 1.4 – Інтерфейс PrimeGate

Недоліки:

- важко інтегрувати сторонні CRM;
- інтерфейс системи відразу здається не дуже зручним;
- є скарги на якість роботи підтримки;
- розрахунок на роботу лише одного фахівця з продажу: вбудована CRM розрахована на одного користувача.

1.3 Аналіз існуючих систем аналітики

На даний момент обсяги електронних даних у світі зростають у геометричній прогресії, що у свою чергу впливає на простоту аналізу цієї

інформації, адже зі збільшенням кількості даних також збільшується складність та потреби щодо покращенні продуктивності апаратного забезпечення.

Не є винятком і така сфера діяльності як банківська справа. Щодня у світі відбуваються мільйони транзакцій, які за певний проміжок часу можуть утворювати собою петабайти даних. Кожна мережа банків прагне покращити свій сервіс, якість обслуговування клієнтів, а на основі власних історичних даних проводити аналіз, який у майбутньому може допомогти з названими вище потребами, а також за допомогою чого можуть прийматися ті чи інші рішення. Наприклад, у статистиці власних клієнтів, тенденції на різні пропозиції тощо. Саме тому, на теперішній час, створення та проектування моделі аналізу даних банківської справи є актуальним рішенням.

Потреби та конкретні умови щодо того, яка саме, яким чином, та з якою метою отримана інформація буде аналізуватися є універсальними для будь-якої області, сфери діяльності, направлення – тому сформувати стовідсоткові критерії та вимоги, які будуть підходящими для всіх не є можливим. Проте, якщо звузити потреби до конкретних цілей, або конкретної сфери діяльності, то можна виділити ті критерії, що будуть релевантними, а також являються загальними та універсальними, які зможуть підійти всім у заданому напрямку.

У даній кваліфікаційної роботі банківська справа була обрана сферою діяльності для подальшого моделювання. У цій області можна виділити два основні набори даних, які використовуються та можуть бути задіяні у аналізі, а також деяку обмежену, або необмежену кількість вторинних джерел отримуваної інформації, що будуть задіяні у подальшому. Двома основними наборами будуть виступати дані транзакцій та дані про користувачів, або іншими словами клієнтів.

Щодо інших обмежених або необмежених джерел даних – це деяка множина довідників або словників, які будуть повинні мати щонайменше два атрибути у своїй структурі – код та опис. Ці довідники можуть бути

використані, по-перше, для побудови вимірів, якщо, наприклад, у даних транзакцій існує атрибут «тип транзакції», що за своєю множиною значень матиме лише кодовий ідентифікатор типу, а словесне представлення цього, зрозуміле для людського ока, буде знаходитися за допомогою довідника. По-друге, такий спосіб організації та формування даних суттєво може зменшити об’єм даних, що будуть передаватися, а також зберігатися, адже немає необхідності записувати та тримати для кожної транзакції її словесний опис типу, якщо це можна завжди віднайти в іншій таблиці, виконавши з’єднання по коду за потреби.

Враховуючи вищесказане, можна сформувати наступні вимоги та критерії, яким має відповідати аналітична Business Intelligence модель, що проектується:

- наявність повного ETL-процесу, що включає в себе завантаження файлів, їх перетворення та приведення до відповідної нормальної форми, з наступним завантаженням у сховище даних;
- завантаження файлів з різних джерел;
- завантаження файлів різного формату;
- валідація вхідних наборів даних – генерувати помилку у випадку невідповідності формату даних, кількості стовпців тощо;
- обробка та очищення вхідного набору даних на можливі помилкові символи у значеннях, наприклад: «№», «@», «#», «$», «%», «^», «&»;
- можливість використання додаткових довідників, які можуть формуватися на клієнтській стороні, що потім будуть використані для встановлення відповідності описів та інших додаткових атрибутів (наприклад, найпростіший варіант – це довідник у форматі «Код – Опис») для відображення цих даних у таблиці фактів, а також у OLAP аналітичній формі даних;
- логування ETL-процесу для його майбутнього відстеження з метою оптимізації проблемних процедур або функцій, що можуть займати багато часу і можуть бути використані більш ефективно;

- синхронізація з історичними чи минулими даними задля можливості простеження тих чи інших тенденцій, проставлення відповідних міток для рядків даних;

- можливість аналізу не тільки теперішніх даних, а й будь-яких історичних, наприклад, за допомогою відповідного фільтру – за замовчуванням відображення тільки теперішніх і актуальних даних, а при вмиканні відповідного фільтру мати можливість перегляду минулих даних;

- можливість аналізу набору даних транзакцій;

- можливість аналізу набору даних акаунтів;

- синхронізація двох датасетів між собою – з’єднання та передача калькульованих полів від рахунків в транзакції і навпаки, щоб мати можливість швидко аналізувати, наприклад, відповідні транзакції конкретного акаунту, не переходячи у інший звіт.

2 ВИБІР ІНСТРУМЕНТІВ ТА ТЕХНОЛОГІЙ ДЛЯ ОБРАНОЇ МОДЕЛІ

2.1 Система керування базами даних Microsoft SQL Server

Microsoft SQL Server – система керування реляційними базами даних (СКБД), що була створена компанією Microsoft [5]. Основна мова, яка використовується у цьому середовищі, мова запитів – Transact-SQL (T-SQL). Ця мова була створена корпорацією Microsoft разом з компанією Sybase. Transact-SQL являє собою структуровану мову запитів SQL з розширеннями за стандартом ANSI/ISO [6]. Дана система керування реляційними базами даних є універсальним рішенням, що може працювати як з домашніми невеликими базами даних, так і з величезними сховищами даних для комерційних цілей, що робить Microsoft SQL Server одним з найбільших конкурентів у цій сфері.

Transact-SQL є процедурним розширенням мови SQL такими можливостями:

- операторами управління;
- локальними та глобальними змінними;
- різноманітними процедурами та методами для обробки будь-яких типів даних: дат, часу, рядків, чисел, математичних обчислень і багато чого іншого;
- автентифікації з використанням облікового запису Microsoft Windows.

Мова Transact-SQL є головною і представляє собою основу для користування Microsoft SQL Server [7]. Всі додатки та розширення, що взаємодіють з цією СУБД, незалежно від їх інтерфейсу, програмного коду та реалізації, відправляють серверу запити та команди на мові Transact-SQL.

В T-SQL присутні спеціальні додаткові команді [8], що надають змогу управляти потоком виконання тих чи інших сценаріїв, керуючи перериваннями, логічними конструкціями, як у звичайних мовах програмування:

- блок групування – структура, що об’єднує процедурний код в логічний одиницю коду: «BEGIN {код} END»;

- блок умови – структура, що виконує перевірку заданої умови та направляє код виконання у відповідному напрямку залежно від перевіреної умови: «IF {умова} {код} ELSE {код}»;

- блок циклу – структура, яка виконує внутрішній логічний код до тих пір, поки виконується задана умова: «WHILE {умова} {код}»;

- оператори управління циклом – виконують переривання всього циклу, або його даної ітерації: «BREAK», «CONTINUE»;

- команда переходу GOTO – оператор, що виконує перехід виконання коду на попередньо оголошену вказану мітку;

- команда затримки WAITFOR – оператор, який виконує затримку у виконанні сценарію;

- виклик помилки RAISERROR – спеціальна команда, яке генерує задану у параметрах помилку. Системні помилки викликаються за таким же принципом [9].

2.2 OLAP-технології

On-Line Analytical Processing (OLAP) – це категорія програмного забезпечення, яка дозволяє користувачам аналізувати інформацію з кількох систем баз даних одночасно. Це технологія, яка дозволяє аналітикам отримувати та переглядати бізнес-дані з різних точок зору.

Аналітикам часто потрібно групувати, агрегувати та об’єднувати дані. Ці операції у інтелектуальному аналізі даних потребують ресурсів. За допомогою OLAP дані можна попередньо обчислити та попередньо агрегувати, що робить аналіз швидшим [10].

Бази даних OLAP розділені на один або кілька кубів. Куби розроблено таким чином, щоб створювати та переглядати звіти стає легко.

В основі концепції лежить куб – це структура даних, яка оптимізована для дуже швидкого аналізу даних. Він складається з числових фактів, які

називаються метриками, що класифікуються за розмірами. Його також називають гіперкубом. На рисунку 2.1 наведена загальна концепція куба OLAP у тривимірному представленні (тому що в ньому може існувати безліч вимірів, які неможливо уявити та зобразити у звичному графічному вигляді).

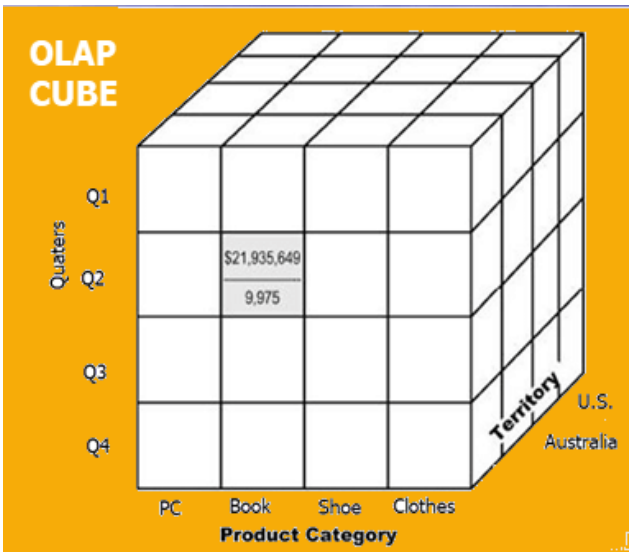


Рисунок 2.1 – Загальне представлення моделі OLAP-куба

Зазвичай операції з даними та аналіз виконуються за допомогою простої електронної таблиці, де значення даних упорядковано у форматі рядків та стовпців. Це ідеальний варіант для двовимірних даних. Однак OLAP містить багатовимірні дані, при цьому дані зазвичай отримують з іншого джерела. Використання електронної таблиці не є оптимальним варіантом. Куб може зберігати та аналізувати багатовимірні дані логічно та впорядковано.

Сховище даних витягуватиме інформацію з багатьох джерел даних і форматів, таких як текстові файли, таблиці Excel, мультимедійні файли тощо. Дані, яки було витягнуто, очищаються та трансформуються. Після вони завантажуються на сервер OLAP (або куб), де інформація попередньо обчислюється для подальшого аналізу.

Нижче наведено основні аналітичні операції OLAP [11]:

- Roll-up – «згорнути» – агрегування по меншому рівню;
- Drill-down – «пробурити» – деталізація будь-якого рівня;

- Slice – «зріз» – у загальному випадку це фільтрування виміру по одному конкретному його члену;

- Dice – «кубик» – подібна до Slice операція, фільтрування відбувається за кількома вимірами;

- Pivot (rotate) – поворот (обертання).

Операція згортання – також відома як «консолідація» або «агрегація». Її можна виконати двома способами:

- зменшення кількості вимірів;

- підйом по концептуальній ієрархії – системі групування речей на основі їх порядку або рівня.

Розглянемо приклад (рисунок 2.2): міста Нью-Джерсі та Лос-Анджелес входять до складу США. Показники продажів у Нью-Джерсі та Лос-Анджелесі становлять 440 та 1560 відповідно. Після операції згортання вони стають 2000.

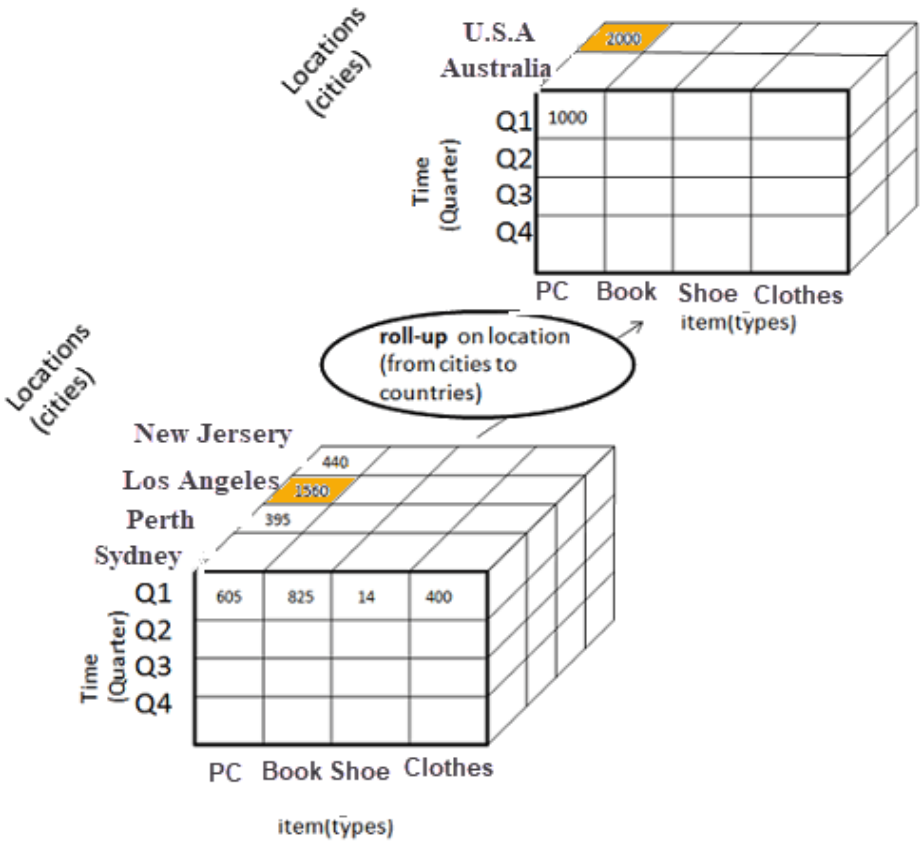


Рисунок 2.2 – Roll-up операція в OLAP

У цьому процесі агрегування ієрархія розташування даних переміщується вгору від міста до країни. У процесі згортання потрібно видалити принаймні один або кілька вимірів. У цьому прикладі параметр «Міста» було видалено.

У разі використання операції Drill-down дані фрагментуються на менші частини. Це протилежність процесу згортання. Його можна зробити через:

- переміщення вниз по концептуальній ієрархії;
- збільшенням кількості вимірів.

Операція Drill-down проілюстрована на рисунку 2.3: квартал Q1 деталізовано до місяців січня, лютого та березня. Відповідні продажі тепер також є для кожного місяця. У цьому прикладі додано місяці вимірювання.

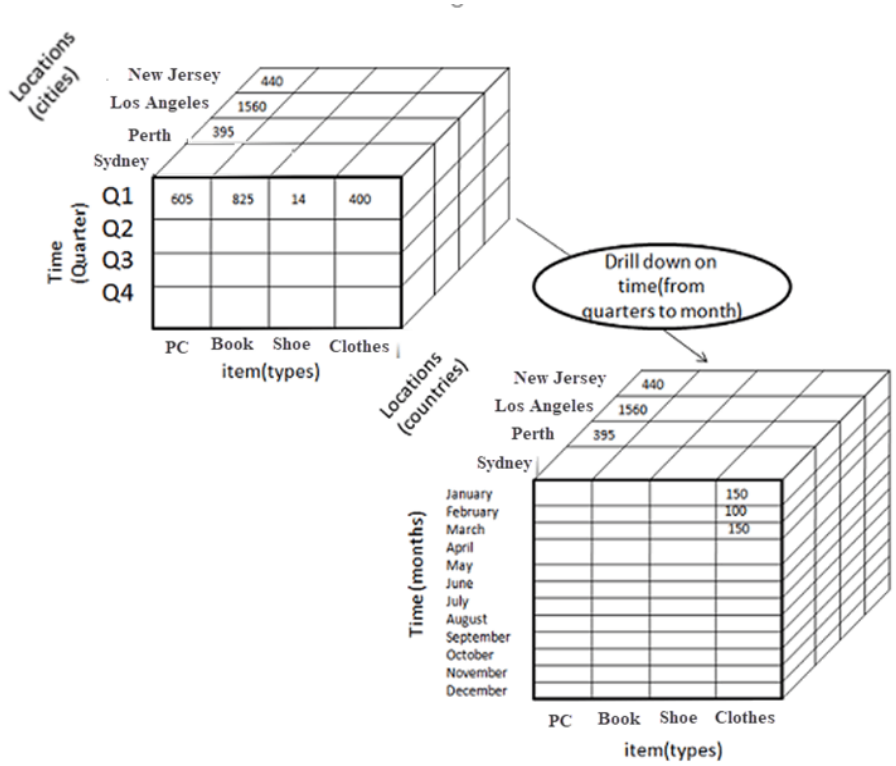


Рисунок 2.3 – Drill-down операція в OLAP

Операція Slice проілюстрована на рисунку 2.4, де обрано один вимір і створено новий підкуб. Вимір часу Time розрізається за допомогою Q1 як фільтра і, як наслідок, взагалі створюється новий куб. Операція Dice схожа на операцію Slice. Різниця в Dice (рисунок 2.4) полягає в тому, що обирається 2 або більше вимірів, які призводять до створення підкуба.

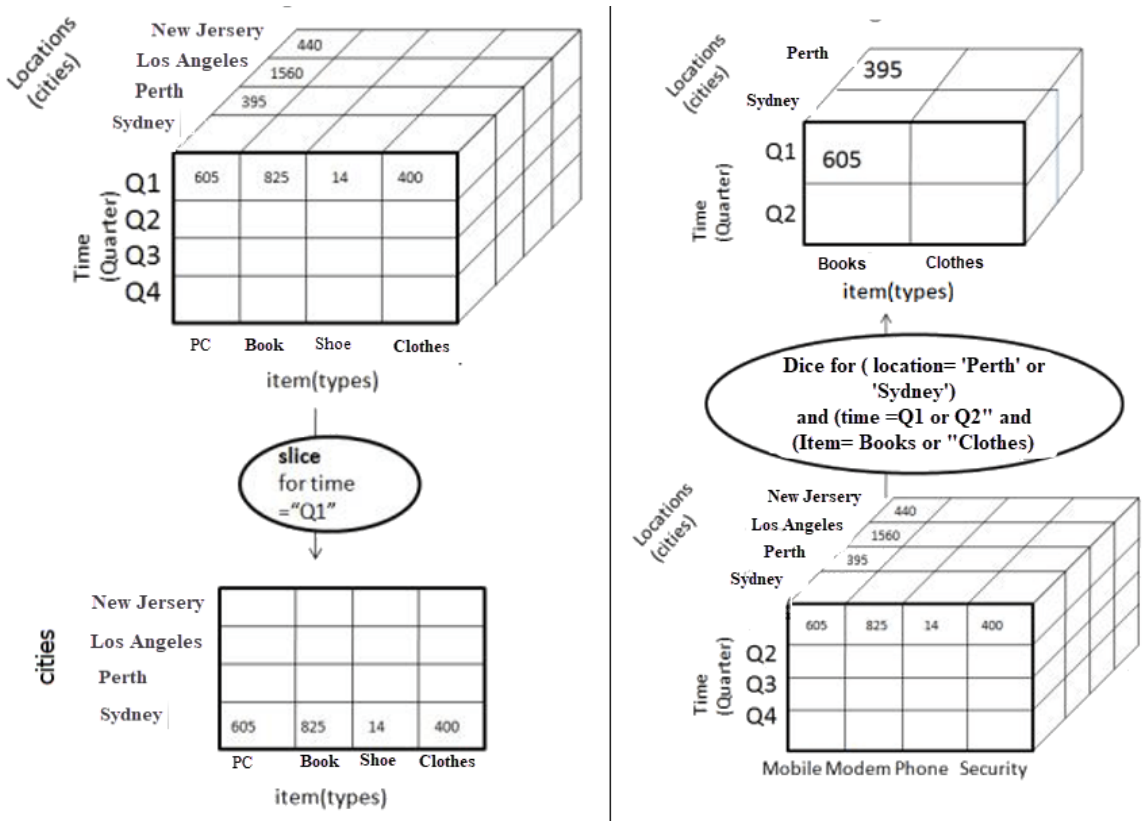


Рисунок 2.4 – Slice (ліворуч) та Dice (праворуч) операції в OLAP

При виконанні операції Pivot відбувається інвертування усіх даних (рядки становляться колонками, а колонки – рядками відповідно), щоб забезпечити альтернативне представлення даних. У прикладі , який наведено на рисунку 2.5, зведена таблиця спочатку базується на типах елементів, а після операції Pivot – на містах.

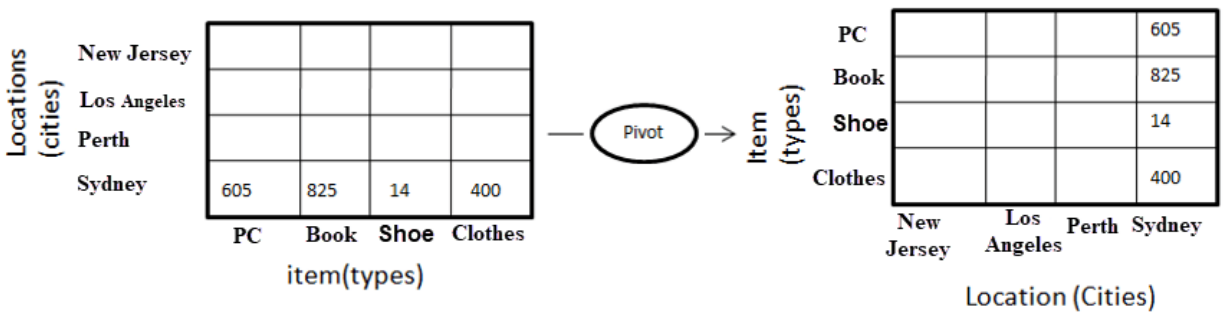


Рисунок 2.5 – Pivot операція в OLAP

Також, на сьогоднішній день, існують різні види OLAP з боку зберігання та агрегування інформації [12], проте розглянемо лише три основні з них, які розміщено на рисунку 2.6.

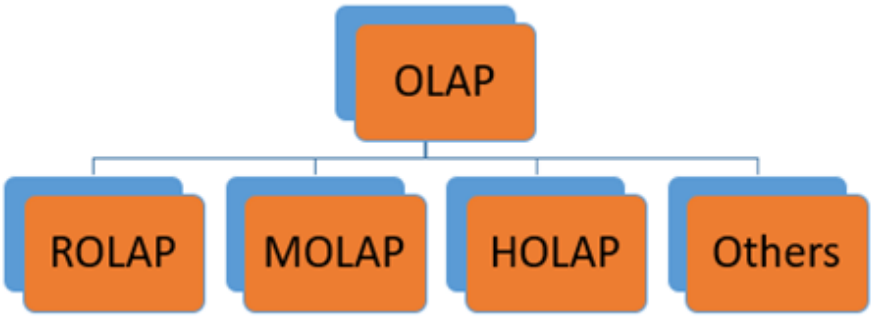


Рисунок 2.6 – Види технологій зберігання OLAP

Технологія ROLAP (Relational On-Line Analytical Processing) – працює з даними, які існують у реляційній базі даних: факти та таблиці вимірювань зберігаються як реляційні таблиці. Вона також дозволяє багатовимірний аналіз даних і на даний момент розвивається найшвидше.

Переваги технології ROLAP:

- висока ефективність даних: пропонує високу ефективність даних, оскільки продуктивність запитів та мова доступу оптимізовані спеціально для аналізу багатовимірних даних;

- масштабованість: цей тип системи OLAP забезпечує масштабованість для керування великими обсягами даних, навіть коли дані постійно збільшуються.

Недоліки технології ROLAP:

- попит на більші ресурси: вимагає високого використання робочої сили, програмного забезпечення та апаратних ресурсів;

- обмеження сукупних даних: інструменти ROLAP використовують SQL для всіх обчислень сукупних даних, однак для обробки обчислень немає встановлених обмежень;

- повільна продуктивність запитів: продуктивність запитів у цій моделі повільна порівняно з іншими технологіями зберігання.

Технологія MOLAP (Multidimensional On-Line Analytical Processing) використовує механізми багатовимірного зберігання на основі масивів для відображення багатовимірних представлень даних. В основному використовують куб OLAP. MOLAP – це класична технологія зберігання для

OLAP, яка полегшує аналіз даних за допомогою багатовимірного куба даних. Дані попередньо обчислюються, повторно підсумовуються та зберігаються в MOLAP (основна відмінність від типу ROLAP). Застосовуючи цю технологію, користувач може використовувати дані багатовимірного перегляду з різними аспектами.

Багатовимірний аналіз даних також можливий, якщо використовується реляційна база даних. Для цього знадобиться запитувати дані з кількох таблиць. Проте MOLAP має всі можливі комбінації даних, які вже зберігаються в багатовимірному масиві, і, як наслідок, є пряма можливість отримати прямий доступ до цих даних. Таким чином, MOLAP є швидшою порівняно з реляційною онлайн-аналітичною обробкою (ROLAP).

Архітектура MOLAP включає такі компоненти [13] (рисунок 2.7):

- сервер бази даних;
- сервер MOLAP;
- інтерфейсний інструмент.

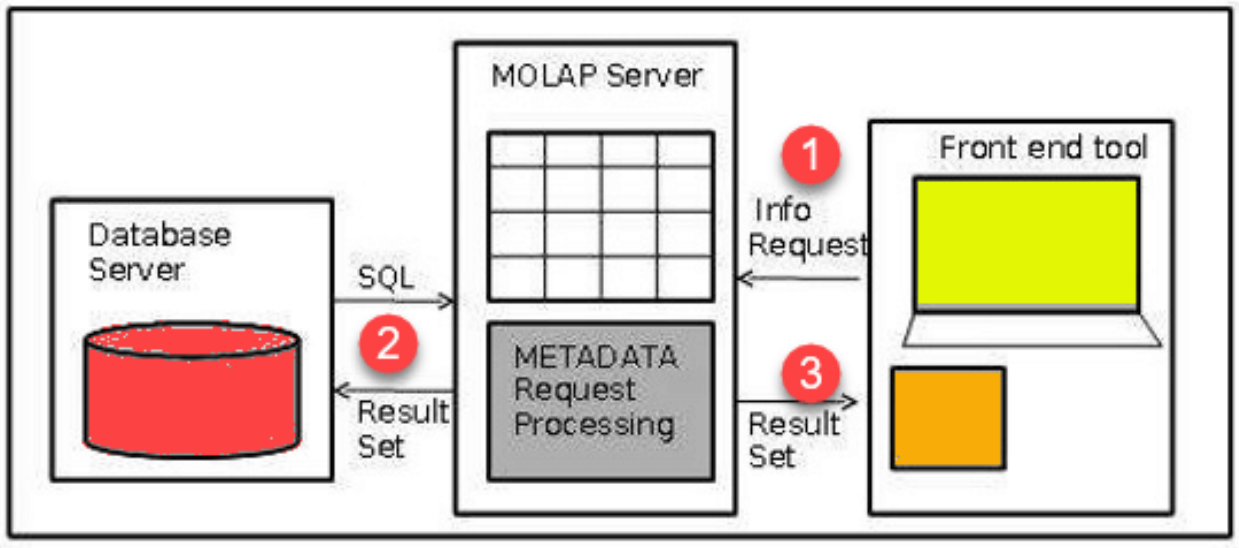


Рисунок 2.7 – Діаграма компонентів MOLAP технології

Розглянемо детальніше кроки на діаграмі, що наведена на рисунку 2.7.

Крок 1: запит користувача надходить через інтерфейс.

Крок 2: рівень прикладної логіки багатовимірної бази даних отримує збережені дані з бази даних.

Крок 3: рівень логіки програми передає результат клієнту/користувачу.

Сервер у MOLAP в основному читає попередньо скомпільовані дані. MOLAP має обмежені можливості для динамічного створення агрегацій або обчислення результатів, які не були попередньо обчислені та збережені.

Наприклад, керівник бухгалтерії може створити звіт, що показує корпоративний рахунок прибутків/збитків або рахунок прибутків/збитків конкретної дочірньої компанії. Багатовимірна база даних у свою чергу буде отримуватиме попередньо скомпільовані цифри та відображатиме цей результат користувачеві.

Ключові моменти в MOLAP:

- операції називаються обробкою;
- інструменти обробляють інформацію з однаковим часом відгуку незалежно від рівня підсумовування;
- інструменти усувають складності розробки реляційної бази даних для зберігання даних для аналізу;
- сервер MOLAP реалізує два рівні представлення зберігання для керування щільними та розрідженими наборами даних;
- використання сховища може бути низьким, якщо набір даних розріджений;
- факти зберігаються в багатовимірному масиві та вимірах, які використовуються для запиту до них.

Зауваження щодо реалізації в MOLAP:

- у MOLAP важливо враховувати як технічне обслуговування, так і зберігання для розробки стратегії створення кубів;
- для запитів до MOLAP використовуються власні мови, наприклад MDX від Microsoft [14];
- API повинні забезпечувати зондування кубів;
- навігація по структурах даних у випадку підтримки кількох предметних областей (коли навігація змінюється, структуру даних необхідно фізично реорганізувати);

- потрібні різні набори навичок та інструменти для адміністратора бази даних для створення та підтримки бази даних.

Переваги MOLAP:

- MOLAP може керувати, аналізувати та зберігати значні обсяги багатовимірних даних;

- висока продуктивність запитів завдяки оптимізованому сховищу, індексуванню та кешуванню;

- менші розміри даних порівняно з реляційною базою даних;

- автоматизоване обчислення вищого рівня агрегатних даних;

- допомагає користувачам аналізувати великі, менш визначені дані;

- MOLAP зручніший для користувача, тому він підходить для недосвідчених користувачів;

- куби MOLAP створені для швидкого пошуку даних і оптимальні для операцій Slice та Dice;

- усі обчислення попередньо генеруються під час створення куба.

Недоліки MOLAP:

- одним із основних недоліків MOLAP є те, що ця технологія є менш масштабованою, ніж ROLAP, оскільки обробляє лише обмежену кількість даних;

- запроваджує резервування даних, оскільки це вимагає ресурсів;

- рішення MOLAP можуть бути тривалими, особливо для великих обсягів даних;

- MOLAP не може містити докладні дані;

- використання сховища може бути низьким, якщо набір даних сильно роздрібнений;

- може обробляти лише обмежену кількість даних, тому неможливо включити велику кількість даних у сам куб.

Технологія HOLAP (Hybrid On-Line Analytical Processing) – є сумішшю технологій ROLAP та MOLAP: пропонується швидке обчислення за допомогою технологій MOLAP та вища масштабованість технологій ROLAP.

Використовує дві бази даних:

- агреговані або обчислені дані зберігаються в багатовимірному кубі OLAP;

- детальна інформація зберігається в реляційній базі даних.

Переваги HOLAP:

- цей вид допомагає економити дисковий простір, а також залишається компактним, що допомагає уникнути проблем, пов'язаних зі швидкістю та зручністю доступу;

- HOLAP використовує технологію куба, яка забезпечує швидшу роботу з усіма типами даних;

- ROLAP миттєво оновлюється, а користувачі HOLAP мають доступ до цих даних, які були миттєво оновлено, у реальному часі. MOLAP забезпечує очищення та перетворення даних, тим самим покращуючи релевантність даних. Це об'єднує найкраще з обох способів.

Недоліки гібридного OLAP:

- вищий рівень складності: основним недоліком технології HOLAP є те, що вона підтримують інструменти й програми як ROLAP, так і MOLAP. Таким чином, це дуже складно;

- потенційні збіги: існує більша ймовірність збігів, особливо щодо їхніх функцій.

Проаналізувавши різні технології зберігання, можна визначити переваги та недоліки OLAP як загального інструменту, незалежно від його виду.

Переваги OLAP:

- OLAP – це платформа для всіх типів бізнесу, включаючи планування, бюджетування, звітування та аналіз;

- інформація та обчислення є узгодженими в кубі OLAP;

- швидко створює та аналізє сценарії «Що, якщо»;

- простий пошук у базі даних OLAP за широкими або конкретними термінами;

- OLAP надає будівельні блоки для інструментів бізнес-моделювання, інтелектуального аналізу даних, звітування про продуктивність;

- дозволяє користувачам розрізати дані за різними параметрами, розмірами та фільтрами;

- OLAP – це добре для аналізу часових рядів;

- за допомогою OLAP легко знайти деякі кластери;

- OLAP – це потужна система онлайн-аналітичного процесу візуалізації, яка забезпечує швидший час відгуку.

Недоліки OLAP:

- OLAP вимагає організації даних у вигляді зірки або сніжинки. Ці схеми складні у впровадженні та адмініструванні;

- користувач не може мати велику кількість вимірів в одному кубі OLAP;

- доступ до даних транзакцій за допомогою системи OLAP неможливий;

- будь-які зміни в кубі OLAP потребують повного оновлення куба. Це трудомісткий процес.

2.3 Microsoft SQL Server Analysis Services

Microsoft Analysis Services – це служби аналізу Microsoft, частина MS SQL (Microsoft SQL Server). Це додатковий набір служб, які пов'язано з бізнес-аналізом та зберіганням даних. Вони включають служби інтеграції (Integration Services) та служби аналізу (Analysis Services). Другі, у свою чергу, містять в собі набір засобів для роботи з інтелектуальним аналізом даних та OLAP технологіями [15]. Також ці інструменти мають скорочену назву SSAS – SQL Server Analysis Services.

Режими зберігання: Microsoft Analysis Services займає нейтральну позицію між MOLAP та ROLAP зберіганням, що дискутується навколо OLAP-продуктів. Завдяки цьому можна використовувати всі види MOLAP, ROLAP та HOLAP усередині однієї моделі.

Режими роздільного зберігання:

- MOLAP: в цьому режимі під обробку потрапляють як самі дані так і їх обробки, які потім індексуються та зберігаються у спеціальному форматі, що був оптимізований під багатовимірні дані;

- ROLAP: в цьому режимі всі дані та їх обробки знаходяться в початковому реляційному джерелі даних, і, як наслідок, зникає необхідність у додатковій обробці;

- HOLAP: дані зберігаються та залишаються у реляційному джерелі даних, а попередні обробки та індекси зберігаються у додатковому форматі, який був оптимізований для зберігання багатовимірних даних.

Режими розмірного зберігання:

- MOLAP: атрибути розмірності зберігаються та обробляються у спеціальному форматі;

- ROLAP: атрибути розмірності залишаються у реляційному джерелі даних, не обробляються. Розділи, що визначаються розмірністю ROLAP, також повинні бути як ROLAP.

Microsoft Analysis Services має інструменти для підтримки різних наборів об'єктних моделей та програмних інтерфейсів (API) для різноманітних операцій у багатьох програмних середовищах [16].

Вилучення даних:

- XML for Analysis – це програмний інтерфейс найнижчого рівня. Він може використовуватися на будь-якій платформі та мові програмування, що мають підтримку HTTP та XML;

- OLE DB for OLAP – розширення OLEDB, заснований на COM та створений для користування у C/C++ програмах;

- ADOMD – розширення ADO, основа – COM Automation. Призначений для Visual Basic програм на платформі Windows;

- ADOMD.NET – розширення ADO.NET, основа – .NET-технологія. Створений для програм, які були написані з використанням управляючого коду на CLR-платформах.

Серед плюсів та переваг SSAS можна виділити такі речі:

- допомагає уникнути конфлікту ресурсів із вхідною системою;

- ідеальний інструмент для чисельного аналізу;

- SSAS дозволяє виявляти шаблони даних, які можуть бути не відразу помітні, використовуючи вбудовані продуктом функції інтелектуального аналізу даних;

- пропонує уніфіковане та інтегроване уявлення всіх запропонованих бізнес-даних: звіти, аналіз показників системи ключових показників ефективності (KPI);

- SSAS пропонує онлайн-аналітичну обробку даних із різних джерел даних.

Все це дозволяє користувачам аналізувати дані за допомогою багатьох інструментів та платформ, включаючи SSRS, а також інтеграцію з офісною програмою для таблиць – Excel.

2.4 Процес завантаження та обробки даних

ETL (Extract-Transform-Load) – це процес, який вилучає дані з різних джерел-систем, потім перетворює дані (наприклад, застосовуючи обчислення, конкатенації, видалення дублікатів, об’єднання даних забезпечуючи якість тощо) і, нарешті, завантажує дані в систему Data Warehouse [17]. Повною формою ETL є Extract, Transform and Load, схему якого наведено на рисунку 2.8.

Легко вважати, що створення сховища даних – це просто вилучення даних із кількох джерел і завантаження в базу даних сховища даних. Це далеко від істини і потребує складного процесу. Процес ETL вимагає активного внеску від різних зацікавлених сторін, включаючи розробників, аналітиків, тестувальників, топ-менеджерів, і є технічно складним.

Інструменти ETL забезпечують стратегії інтеграції даних, дозволяючи компаніям збирати дані з багатьох джерел даних та консолідувати їх в одному централізованому місці. Інструменти ETL також уможливлюють

спільну роботу різних типів даних.

Щоб зберегти свою цінність як інструменту для осіб, що приймають рішення, система сховища даних має змінюватися разом зі змінами в бізнесі. ETL – це діяльність системи сховища даних, що повторюється (щодня, щотижня, щомісяця). І вона має бути гнучкою, автоматизованою та добре задокументованою.

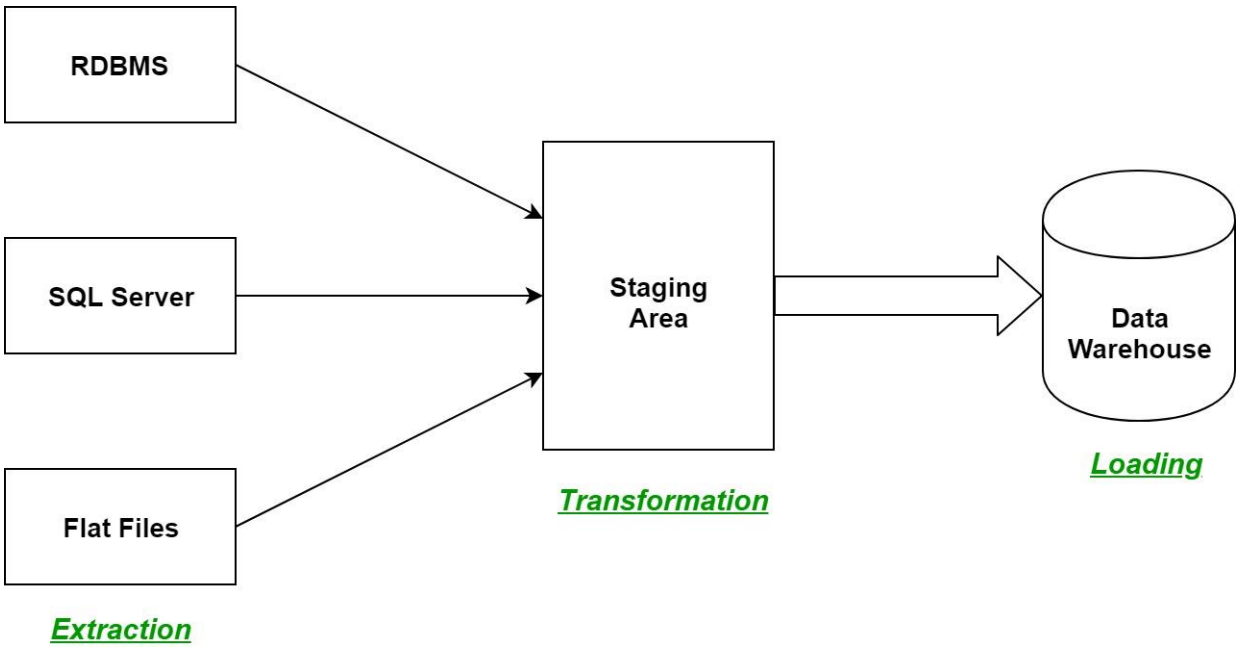


Рисунок 2.8 – Схематичне зображення ETL-процесу

Існує багато причин для впровадження ETL в організації:

- допомагає компаніям аналізувати свої бізнес-дані для прийняття важливих бізнес-рішень;

- транзакційні бази даних не можуть відповісти на складні бізнес-питання, на які можна відповісти на прикладі ETL;

- Data Warehouse забезпечує загальне централізоване сховище даних.

Інструменти ETL також дають змогу переміщувати дані між різними джерелами, призначеннями та інструментами аналізу. Як наслідок, процес ETL відіграє вирішальну роль у створенні бізнес-аналітики та реалізації більш широких стратегій управління даними [18].

У міру зміни джерел даних сховище даних (Data Warehouse)

автоматично оновлюватиметься. Добре розроблена та задокументована система ETL необхідна для успіху проекту сховища даних. Також впроваджується перевірка правил перетворення, агрегацій та обчислень даних.

Процес ETL:

- дозволяє порівнювати вибіркові дані між вхідною та цільовою системами;

- може виконувати складні перетворення та потребує додаткової області для зберігання даних;

- допомагає перенести дані в сховище даних. Перетворює в різні формати та типи, щоб дотримуватися єдиної системи.

ETL – це попередньо визначений процес для доступу та обробки вихідних даних у цільовій базі даних. У сховищі даних він пропонує глибокий історичний контекст для бізнесу, дозволяє поєднувати застарілі корпоративні дані з даними, зібраними з нових платформ і програм. Це створює довгостроковий перегляд даних, щоб можна було переглядати старіші набори даних разом із новішою інформацією.

Критичні компоненти ETL – незалежно від конкретного процесу, який можна обрати, є деякі важливі компоненти, що потрібно розглянути:

- підтримка збору змінених даних (CDC) (відома також як реплікація binlog): поступове завантаження дозволяє оновлювати аналітичне сховище новими даними без повного перезавантаження всього набору даних;

- аудит та ведення журналів: потрібне детальне ведення журналів у конвеєрі ETL, щоб забезпечити можливість перевірки даних після їх завантаження та усунення помилок;

- відмовостійкість: у будь-якій системі неминуче виникають проблеми. Системи ETL повинні мати можливість акуратно відновлюватися, забезпечуючи можливість проходження даних від одного кінця конвеєра до іншого, навіть якщо під час першого запуску виникають проблеми;

- підтримка сповіщень: щоб організація довіряла своїм аналізам,

повинні бути створені системи, які сповіщатимуть, коли дані є неточними. Наприклад, якщо в конекторі сталася неочікувана помилка, автоматично створюється заявка, щоб її перевірив інженер;

- використання моніторингу системного рівня для таких речей, як помилки в мережі або базах даних;

- низька затримка: деякі рішення потрібно приймати в режимі реального часу, тому актуальність даних має вирішальне значення. Хоча існуватимуть обмеження затримки, накладені певною інтеграцією вихідних даних, дані мають проходити через процес ETL із якомога меншою затримкою;

- масштабованість: із зростанням компанії зростатиме й обсяг даних. Усі компоненти процесу ETL мають масштабуватися, щоб підтримувати довільно велику пропускну здатність [19];

- точність: дані не можна викидати або змінювати таким чином, щоб спотворити їх значення. Кожну точку даних слід перевіряти на кожному етапі процесу.

3 МОДЕЛЬ БАЗИ ДАНИХ ТА OLAP-КОМПОНЕНТУ, ЩО РОЗРОБЛЯЮТЬСЯ

3.1 Загальний огляд моделі

Опираючись на сформовані вимоги, яким має відповідати аналітична Business Intelligence модель, що проектується, потрібно врахувати та реалізувати кожен критерій. Модель, що розробляється, повинна мати повний ETL-процес, який включає в себе завантаження файлів з різних джерел та різних форматів, їх перетворення та приведення до відповідної нормальної форми, а потім завантаження у сховище даних. Під час етапу «Extract» повинна відбуватися валідація вхідних даних та генеруватися помилка у випадку невідповідності формату даних, кількості стовпців тощо. На етапі «Transformation» – обробка та очищення вхідного набору даних на можливі помилкові символи у значеннях.

Також повинна бути передбачена можливість використання додаткових довідників, які можуть формуватися на клієнтській стороні, що потім будуть використані для встановлення відповідності описів та інших додаткових атрибутів (наприклад, найпростіший варіант – це довідник у форматі «Код – Опис») для відображення цих даних у таблиці фактів, а також у OLAP аналітичній формі даних.

Під час будь-якого етапу ETL-процесу має місце його логування для майбутнього відстеження з метою оптимізації проблемних процедур або функцій, що можуть займати багато часу і можуть бути використані більш ефективно [20].

Невід’ємною частиною Business Intelligence є синхронізація з історичними чи минулими даними задля можливості простеження тих чи інших тенденцій, проставлення відповідних міток для рядків даних. Це включає в себе можливість аналізу, наприклад, за допомогою відповідного фільтру – за замовчуванням відображення тільки теперішніх та актуальних

даних, а при вмиканні відповідного фільтру мати можливість перегляду минулі даних.

Так як мається два датасети (акаунти та транзакції), то буде розроблено дві факт-таблиці відповідно, які у свою чергу будуть синхронізовані між собою – з'єднання та передача калькульованих полів від рахунків в транзакції та навпаки, щоб мати можливість швидко аналізувати, наприклад, відповідні транзакції конкретного акаунту, не переходячи у інший звіт.

На рисунку 3.1 наведено ссхематичне зображення розробленого ETL-процесу для моделі кваліфікаційної роботи.

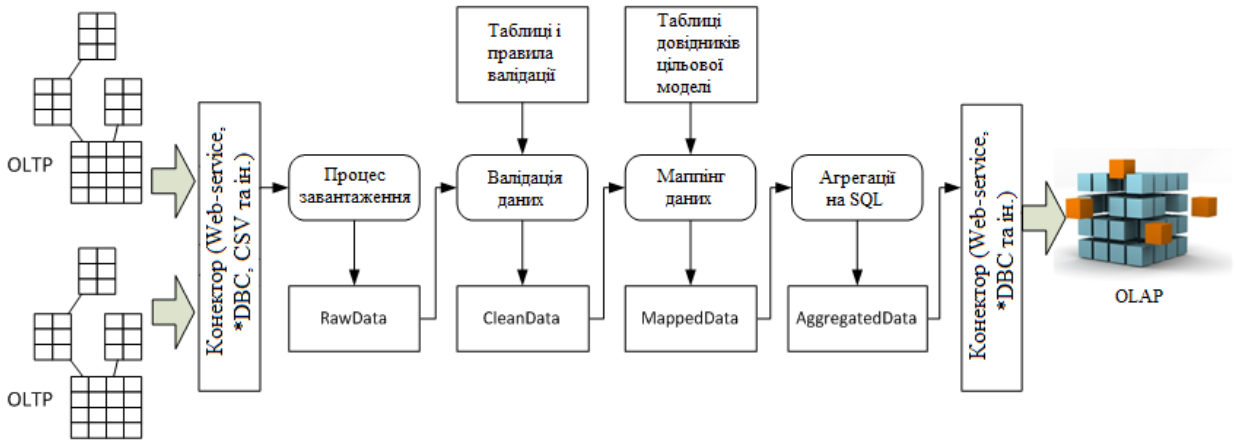


Рисунок 3.1 – Схематичне зображення ETL-процесу

Процес завантаження: на цьому етапі всі вхідні дані (попередньо визначені) потрапляють у базу даних до raw-таблиць для подальшої обробки. Також на цьому етапі здійснюється логування завантажених файлів, перевірка сум отриманих файлів (наприклад, у файлі мається останній рядок зі значенням кількості рядків у файлі, а також контрольними сумами по тим чи іншим атрибутам – ці значення повинні співпадати зі значеннями, що були вирахувані з боку базу даних на основі raw-таблиці).

Валідація даних: задачею цього етапу є послідовна перевірка на повноту та коректність даних, очистка від небажаних символів у значеннях даних (наприклад: «№», «@», «#», «$», «%», «^», «&»), логування помилок для генерації звіту валідації у подальшому.

Маппінг даних: на цьому етапі до очищених таблиць додаються

атрибути з довідників. Значення цих атрибутів можуть бути проставлені як один до одного, багато до одного, так і багато до багатьох. У випадку використання двох останніх варіантів потрібні налаштування в ETL-інструменті, реалізація коду та логіки, за якими будуть проставлені відповідні значення.

Агрегації на SQL – це крок, на якому відбуваються обчислення на рівні бази даних, що будуть використовуватися у детальному відображенні даних, наприклад, сума всіх транзакцій для акаунту буде збережена у даних акаунту, а ідентифікаційний номер рахунку буде розповсюджений між всіма його транзакціями, щоб мати змогу швидко аналізувати їх на рівні ідентифікаційного номеру без потреби постійної перевірки с датасетом акаунтів.

Після того, як всі обробки на стороні бази даних завершено, обчислення зі свого боку починає OLAP-компонент. Загальний термін оновлення даних в ньому має назву процесінг (processing) – набір дій, при яких відбувається вираховування агрегацій для факт-таблиць, оновлення властивостей та атрибутів вимірів, дії над партиціями та властивостями кубів тощо.

Крок 1: потрібно оновити колекції даних та атрибутів у вимірах, адже під час ETL-процесу таблиці довідників, що виступають у ролі джерел для вимірів, могли бути доповнені новими рядками або оновлені атрибути уже існуючих (довідник повинен містити в собі повний набір унікальних значень ключів вимірів, що зустрічаються в даних, інакше під час розрахунку кубу виникне помилка через неповноцінність даних).

Крок 2: залежно від налаштованого механізму, створити для нових рядків, що були завантажені до сховища даних так названі партиції – розріз даних, який фізично розділяє дані в кубі на частини. Це є важливим кроком з боку оптимізації, адже якщо історичні дані не оновлюються, то немає потреби виконувати перерахування агрегацій для них – тому розділення даних на історичні (уже присутні раніше дані, що не змінювалися, можливе

більш детальне розділення всередині цієї частини даних) та поточні (нові дані) партиції на цьому кроці є важливим.

Крок 3: новостворені партиції повинні бути оброблені, а також обчислені агрегації для них – калькуляції та метрики на всіх можливих точках перетину будь-яких визначених у кубах вимірів.

Виходячи з вищесказаного, можна сформувати інформаційне представлення моделі, що розробляється. Найвищий рівень системи поділяється на дві основні частини та представляє собою подану рівність (3.1):

$$M = \{db, olap\}, \tag{3.1}$$

де db (database) – множина елементів об’єктів бази даних;

olap (on-line analytical processing) – множини елементів об’єктів OLAP.

Кожен з поданих складових елементів у свою чергу може бути розділений на окремі частини, модулі, логічні складові. Таким чином, об’єкти бази даних можуть бути представлені як множина внутрішніх постійних елементів для організації ETL-процесу та сукупність даних, що звісно буде змінюватись (3.2):

$$db = \{ipo, d\} \tag{3.2}$$

де ipo (internal permanent objects) – множина внутрішніх постійних елементів для організації ETL-процесу;

d (data) – сукупність даних.

Складові внутрішніх елементів представлені як набори функцій, процедур, тригерів, зовнішніх збірок (assemblies), а також додаткових таблиць та представлень для роботи перерахованих раніше елементів утворюють окремі множини $\{ipo_i, i = 0, 1, \dots, N\}$, де i – номер множини внутрішніх елементів, N – кількість потоків незалежних даних. Під i=0 мається на увазі набір складових, що є загальними та універсальними, які

будуть існувати в базі даних навіть після кардинальних змін, і можуть бути використаними для будь-якого потоку даних, незалежно від його властивостей, наприклад, процедура логування дій ETL-процесу – будуть змінюватися лише параметри, з якими вона буде викликатися (назва частини процесу, час початку, додаткові параметри тощо). Будь-яку множину внутрішніх елементів можна представити як кортеж, поданий нижче (3.3):

$$ipo_i = \{pr_i, f_i, tr_i, asm_i, avw_i, at_i\}, \quad (3.3)$$

де pr_i (procedures) – процедури;

f_i (functions) – функції;

tr_i (triggers) – тригери;

asm_i (assemblies) – зовнішні збірки;

avw_i (views) – додаткові представлення;

at_i (additional tables) – додаткові таблиці.

Щодо сукупності даних, вони в свою чергу поділяються на окремі множини датасетів $\{d_j, j = 1, 2, \dots, N\}$, де j – номер множини конкретних датасетів, N – кількість потоків незалежних даних. Кожен з цих датасетів складається з окремих елементів, що поділені по етапам обробок, довідникам та сховищам даних (3.4):

$$d_i = \{raw_i, st_i, fct_i, lkp_i\} \quad (3.4)$$

де raw_i – raw-таблиця (є першою, до якої надходять дані при Extract етапі);

st_i (stage) – stage-таблиця (використовується на етапі Transform, дані надходять з raw-таблиць);

fct_i (fact) – fact-таблиця (використовується для зберігання даних і являє собою фінальне сховище даних і джерело для OLAP системи);

lkp_i (lookup) – множина таблиць довідників.

Виходячи з проведеного аналізу, модель OLAP компоненту можна

поділити на фізичні та логічні складові та представити у вигляді (3.5):

$$olap = \{dm, mt, clc, prt\} \quad (3.5)$$

де dm (dimension) – множина вимірів у OLAP-кубі;

mt (metric) – множина метрик;

clc (calculation) – множина калькуляцій;

prt (partition) – множина партицій.

Таким чином, можна сформувати загальну інформаційну рівність моделі, опираючись на визначені представлення частин моделі, схема яких подана на рисунку 3.2.

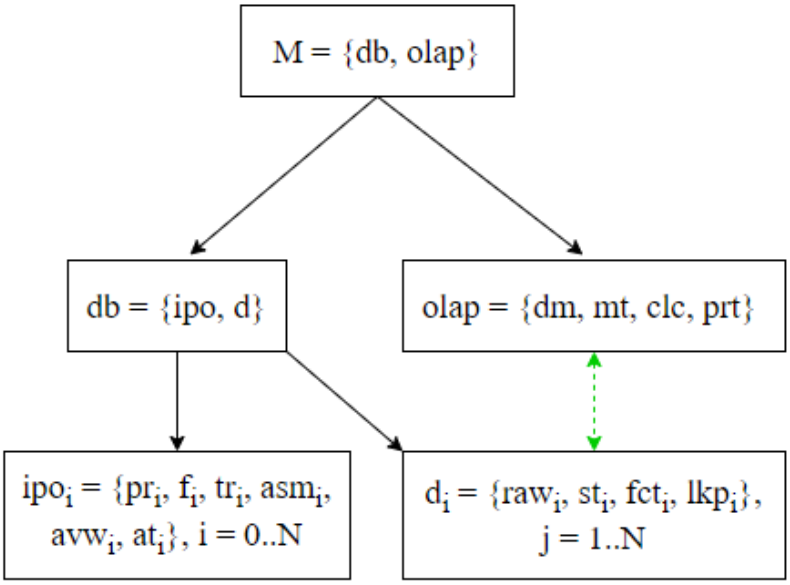


Рисунок 3.2 – Схематичне зображення елементів моделі

3.2 Аналіз структури даних транзакцій та рахунків

Як вже було описано попередньо, у наборі вхідних даних будуть присутні два основні датасети – транзакції та рахунки (або іншим словом – акаунти). Перший набір описує будь-які платежі, оплати, зарахування, переміщення коштів «з» та «на» рахунки-акаунти. Другий набір даних описує клієнтів – фізичних осіб, що користуються послугами банку. Тут міститься

інформація про ідентифікаційні дані людини, її активні послуги, баланс тощо. Для детального аналізу розглянемо скрипт для створення raw-таблиці транзакцій, що наведено у лістингу 3.1.

Лістинг 3.1 – Скрипт створення raw-таблиці транзакцій

```
CREATE TABLE _rawTXNS (
     RowID INT IDENTITY(1,1) NOT NULL,
     AccountNumber VARCHAR(100) NULL,
     Department VARCHAR(100) NULL,
     TransactionSentDate VARCHAR(100) NULL,
     TransactionSentTime VARCHAR(100) NULL,
     TransactionCompletionDate VARCHAR(100) NULL,
     TransactionCompletiontime VARCHAR(100) NULL,
     TxnPaymentPlan VARCHAR(100) NULL,
     TxnID VARCHAR(100) NULL,
     TxnAmount VARCHAR(100) NULL,
     TxnType VARCHAR(100) NULL,
     TxnCategory VARCHAR(100) NULL,
     Poster VARCHAR(100) NULL,
     Place VARCHAR(100) NULL,
     Address VARCHAR(100) NULL,
     CurrentBalance VARCHAR(100) NULL,
     SuccessfullFlag VARCHAR(100) NULL,
     ReasonOfRollback VARCHAR(100) NULL,
     ReceiverIBANNumber VARCHAR(100) NULL,
     Currency VARCHAR(100) NULL,
     PaymentPurpose VARCHAR(100) NULL,
     TaxPercent VARCHAR(100) NULL,
     TaxAmount VARCHAR(100)
)
```

Розглянемо кожне поле даних:

- RowID – внутрішнє службове поле, що має цілочисельний тип даних з властивістю IDENTITY, яке представляє собою лічильник. Використовується для унікальної нумерації рядків у таблиці, і в якості первинного ключа;

- AccountNumber – номер рахунку-акаунту, унікально ідентифікує клієнта банку;

- Department – ідентифікатор відділення банку, у випадку проведення транзакцій у касах;

- TransactionSentDate – дата створення та відправлення транзакції на обробку;

- TransactionSentTime – час створення та відправлення транзакції на

обробку;

- TransactionCompletionDate – дата успішного опрацювання транзакції;

- TransactionCompletionTime – час успішного опрацювання транзакції;

- TxnPaymentPlan – тарифний план, який був застосований до транзакції (наприклад, впливає на відсоток комісії);

- TxnID – унікальний ідентифікатор транзакції, що представляє собою стандартизований UUID (за стандартом ISO/IEC 11578:1996);

- TxnAmount – сума, чисельне значення якої означає кількість коштів, що було передано;

- TxnType – тип транзакції, який може мати два значення: «Credit» та «Debit». Виходячи з наявних значень, це напрямок транзакції – списання або зарахування коштів;

- TxnCategory – категорія транзакції, більш детальний розріз типу транзакції, наприклад, зарахування через переказ, або зарахування через поповнення з терміналу тощо;

- Poster – спосіб здійснення операції, який визначається, при здійсненні операцій через веб-банкінг, мобільний додаток, безконтактну оплату;

- Place – місце проведення транзакції, що представляє собою код закладу, куди була направлена транзакція;

- Address – місцезнаходження проведення транзакції, визначається за можливості;

- CurrentBalance – залишок на рахунку після проведення транзакції;

- SuccessfullFlag – булеве значення, за яким встановлюється, чи була транзакція успішною;

- ReasonOfRollback – причина помилки або повернення транзакції;

- ReceiverIBANNumber – IBAN-номер отримувача;

- Currency – валюта, у якій була проведена транзакція;

- PaymentPurpose – призначення платежу, коментар відправника;

- TaxPercent – відсоток комісії, яка була списана (не враховується в розмірі транзакції);

- TaxAmount – абсолютне значення комісії.

Другим датасетом виступає набір даних рахунків клієнтів. Розглянемо скрипт для створення raw-таблиці акаунтів, що наведено у лістингу 3.2.

Лістинг 3.2 – Скрипт створення raw-таблиці акаунтів

```
CREATE TABLE _rawAccounts (
     RowID INT IDENTITY(1,1) NOT NULL,
     AccountNumber VARCHAR(100) NULL,
     AccountSuffix VARCHAR(100) NULL,
     TaxIDNumber VARCHAR(100) NULL,
     RegistrationDepartment VARCHAR(100) NULL,
     Registrar VARCHAR(100) NULL,
     RegistrationDate VARCHAR(100) NULL,
     Address VARCHAR(100) NULL,
     PhoneNumber VARCHAR(100) NULL,
     City VARCHAR(100) NULL,
     ClientFirstName VARCHAR(100) NULL,
     ClientLastName VARCHAR(100) NULL,
     ClientMiddleName VARCHAR(100) NULL,
     ClientDOB VARCHAR(100) NULL,
     PaymentPlan VARCHAR(100) NULL,
     CurrentBalance VARCHAR(100) NULL,
     CurrentCreditFlag VARCHAR(100) NULL,
     CreditStartDate VARCHAR(100) NULL,
     CreditEndDate VARCHAR(100) NULL,
     CreditDuration VARCHAR(100) NULL,
     CreditAmount VARCHAR(100) NULL,
     CreditPercent VARCHAR(100) NULL,
     CurrentCreditBalance VARCHAR(100) NULL,
     LastCreditPaymentDate VARCHAR(100) NULL,
     ARStatus VARCHAR(100) NULL,
     AccountClosingDate VARCHAR(100) NULL,
     IBANNumber VARCHAR(100) NULL,
     MainCurrency VARCHAR(100) NULL,
     PiggyBankFlag VARCHAR(100) NULL,
     PiggyBankNumber VARCHAR(100) NULL,
     PiggyBankBalance VARCHAR(100) NULL,
     PiggyBankPercent VARCHAR(100) NULL,
     PiggyBankStartDate VARCHAR(100) NULL,
     PiggyBankEndDate VARCHAR(100) NULL
)
```

Дана таблиця включає в себе такі поля:

- RowID – внутрішнє службове поле, що має цілочисельний тип даних з властивістю IDENTITY, що представляє собою лічильник. Використовується для унікальної нумерації рядків у таблиці, і в якості первинного ключа;

- AccountNumber – номер рахунку-акаунту, який унікально ідентифікує клієнта банку;

- TaxIDNumber – ідентифікаційний код особи;

- RegistrationDepartment – відділення, де було зареєстровано рахунок;

- Registrar – працівник банку, який зареєстрував рахунок;

- RegistrationDate – дата реєстрації рахунку;

- Address – адреса особи, яка відкриває рахунок;

- PhoneNumber – мобільний телефон особи, яка відкриває рахунок;

- City – місто клієнта;

- ClientFirstName – ім’я клієнта;

- ClientLastName – прізвище клієнта;

- ClientMiddleName – по-батькові клієнта;

- ClientDOB – дата народження клієнта;

- PaymentPlan – поточний тарифний план відкритого рахунку;

- CurrentBalance – поточний баланс;

- CurrentCreditFlag – значення «Так» або «Ні», що означає, чи має даний клієнт відкритий кредит;

- CreditStartDate – дата початку кредитного договору;

- CreditEndDate – дата закінчення кредитного договору;

- CreditDuration – тривалість кредитного договору;

- CreditAmount – значення, на яку суму був виданий кредит;

- CreditPercent – відсоток, під який був виданий кредит;

- CurrentCreditBalance – поточний баланс кредитного рахунку;

- LastCreditPayment – дата останнього платежу за кредитом;

- ARStatus – загальний поточний стан акаунту (активний або неактивний);

- AccountClosingDate – у випадку неактивного статусу рахунку його дата закриття;

- IBANNumber – IBAN-номер поточного рахунку;

- MainCurrency – основна валюта для акаунту;

- PiggyBankFlag – булеве значення, що означає наявність «Скарбнички» на цьому рахунку;

- PiggyBankNumber – унікальний ідентифікатор «Скарбнички»;

- PiggyBankBalance – поточний баланс «Скарбнички»;

- PiggyBankPercent – відсоток річних «Скарбнички»;

- PiggyBankStartDate – дата початку договору «Скарбнички»;

- PiggyBankEndDate – дата закінчення договору «Скарбнички».

Наведені raw-таблиці – лише перший та початковий етап всього ETL-процесу, а саме частини «Extract». Далі будуть проведені наступні кроки: перевірки, очистки, фільтрування, з'єднання двох датасетів між собою – передача агрегованої інформації з транзакцій в акаунти та навпаки, а також фінальний крок – завантаження у кінцеві таблиці, що далі будуть слугувати джерелом даних для OLAP-системи розроблюваної моделі.

3.3 Проектування моделі бази даних

Як вже було описано вище, база даних повинна мати свій ETL-процес для завантаження та обробки даних. В загальному вигляді, процедури на мові Transact-SQL представляють собою послідовність дій та команд як у звичайних мовах програмування, тому потрібно розділяти три стадії ETL-процесу – Extract, Transform та Load на логічні блоки за допомогою процедур, що будуть виконуватися послідовно один за одним для окремого набору даних. У випадку декількох потоків завантаження даних, процеси можуть бути паралельними один до одного – наприклад, якщо один датасет перебуває на етапі Extract, то інший в цей час вже може виконуватися на етапі Load. Для реалізації процесу логування ETL-процесу, потрібно для початку створити таблицю, яку буде зберігати в собі ці записи (лістинг 3.3).

Лістинг 3.3 – Створення таблиці для логування

```
CREATE TABLE logETL (
     ID INT IDENTITY(1,1) NOT NULL,
     Activity VARCHAR(1000) NOT NULL,
```

```
    StartDate DATETIME NOT NULL,
    Duration VARCHAR(100) NULL,
    Status VARCHAR(100) NOT NULL,
    ProcessedRows INT NULL,
    CONSTRAINT PK_logETL PRIMARY KEY (ID)
)
```

Дана таблиця включає в себе такі поля:

- ID – поле, що має цілочисельний тип даних з властивістю IDENTITY, що представляє собою лічильник. Використовується для унікальної нумерації рядків у таблиці, а також у якості первинного ключа;

- Activity – словесний опис операції, що була залогована;

- StartDate – дата та час початку операції;

- Duration – тривалість операції, що виконується. На початку матиме значення NULL, буде оновлено при завершенні відповідних дій;

- Status – статус операції. На початку матиме значення «Started», буде оновлено на «Success» при завершенні відповідних дій;

- ProcessedRows – у випадку наявності, буде записано значення оброблених рядків, наприклад, якщо виконувана процедура проводить дії над однією таблицею – DELETE, UPDATE, INSERT, тощо.

CONSTRAINT PK_logETL – це первинний ключ, що побудований на основі поля ID, має лише унікальні значення так як поле є лічильником. Внаслідок цього, на ньому буде побудований кластерний індекс, за яким таблиця буде відсортована у пам’яті. Це допоможе виконувати швидкі та оптимальні операції з’єднання у випадку використання поля ID в цій умові.

Наступний крок щодо логування: створення відповідної процедури, яка буде викликатися на початку тієї чи іншої частини коду, а також в кінці – для встановлення тривалості виконання визначеної частини коду, встановлення успішного/неуспішного статусу і, за наявності, кількості оброблених рядків. Код процедури наведений у лістингу 3.4.

Лістинг 3.4 – Реалізація процедури логування usp_WriteLog

```
CREATE OR ALTER PROCEDURE usp_WriteLog
    @Activity VARCHAR(1000),
```

```
     @Status VARCHAR(10),
     @ProcessedRows INT = NULL
AS
BEGIN
     IF @Status = 'Started'
     BEGIN
          INSERT INTO logETL (StartDate, Activity, Status)
          VALUES(GETDATE(), @Activity, @Status)
     END

     ELSE IF @Status = 'Succeed'
     BEGIN
          DECLARE @seconds INT

          UPDATE logETL SET
          @seconds = DATEDIFF(SECOND, StartDate, GETDATE()),
          Duration = IIF(@seconds < 86400,
               CONVERT(VARCHAR(12), DATEADD(SECOND, @seconds, 0),
 108), ' > 1 day'), Status = @Status, ProcessedRows =
@ProcessedRows
          WHERE ID = (SELECT TOP 1 ID FROM logETL
          WHERE Activity = @Activity ORDER BY ID DESC)
     END
END
GO
```

Розглянемо докладніше алгоритм процедури usp_WriteLog.

Крок 1. Якщо вхідний параметр @Status має значення «Started», то означає, що логування викликається на початку потрібної частини коду і потрібно зробити новий запис у відповідній таблиці логів – виконати операцію INSERT в таблицю logETL, параметром StartDate для якої буде поточна дата (функція GETDATE() повертає тип DATETIME зі значенням поточної дати та часу).

Крок 2. У іншому випадку, якщо вхідний параметр @Status має значення не «Started», а «Succeed», то це означає що виконання конкретного коду завершено і потрібно залогувати його завершення, з вирахуванням тривалості та кількості оброблених рядків. Потрібно оновити рядок у таблиці logETL з тією ж назвою Activity, що є вхідним параметром, а саме обрати останній рядок з відповідним значенням колонки Activity, та вилучити з нього значення ID, щоб оновити рядок з цим значенням, бо ID є унікальним ідентифікатором рядку. Оновлення відбувається для колонок:

- Duration – це різниця в секундах між значеннями StartDate та поточною датою, а у випадку цієї різниці більше 24 годин (86400 секунд), записати « > 1 day»;

- Status – це заміна значення на «Succeed», так як операція була завершена успішно;

- ProcessedRows – це кількість оброблених рядків, зазвичай після операцій DELETE, UPDATE, INSERT.

Для конкретної частини коду процедура логування буде викликатися два рази – на початку (лістинг 3.5) та в кінці (лістинг 3.6) для оновлення даних, за допомогою команди T-SQL «EXEC».

Лістинг 3.5 – Приклад виклику логування на початку

```
DECLARE @Activity VARCHAR(1000) = OBJECT_NAME(@@PROCID)
EXEC usp_WriteLog @Activity, 'Started'
```

де OBJECT_NAME() – це системна функція отримання ім’я об’єкту, що приймає в параметри значення ID об’єкту;

@@PROCID – це системна змінна, що має ID-значення поточної процедури, що виконується, у якій відбулося звернення.

Лістинг 3.6 – Приклад виклику логування в кінці

```
EXEC usp_WriteLog @Activity, 'Succeed', @@ROWCOUNT
```

У цьому випадку використовується системна змінна @@ROWCOUNT, що зберігає в собі значення кількості останніх оброблених рядків в області видимості, де вона була викликана.

Також, окрім логування процедур ETL-процесу, є доцільним зберігання інформації про всі файли, що завантажуються (лістинг 3.7).

Лістинг 3.7 – Створення таблиці інформації про файли

```
CREATE TABLE logImport (
     ID INT IDENTITY(1,1) NOT NULL,
     ImportDateTime DATETIME DEFAULT GETDATE(),
     RawTableName VARCHAR(100) NULL,
     RawFileName VARCHAR(100) NULL,
```

```
    RawFileRows INT,
    RawFileAmount MONEY,
    FormatFileName VARCHAR(100) NULL,
    PostDate DATETIME NULL,
    PostDate_ID INT NOT NULL,
    ProcessedRows INT NOT NULL DEFAULT 0,
    CONSTRAINT PK_logImport PRIMARY KEY (ID)
)
```

Відповідно таблиця має наступні атрибути:

- ID – ціле число, має властивість IDENTITY – це лічильник. Використовується для унікальної нумерації рядків у таблиці, а також у якості первинного ключа (вказано в CONSTRAINT PK_logImport);

- ImportDateTime – час та дата завантаження файлу в базу даних;

- RawTableName – назва цільової raw-таблиці, до якої повинні потрапити дані з відповідного файлу;

- RawFileName – назва файлу, який повинен бути завантажений;

- RawFileRows – кількість рядків у файлі, що завантажується. Вирахування цього відбувається за допомогою зовнішньої збірки (assembly), яка написана на мові C# (лістинг 3.8);

- RawFileAmount – контрольне значення, сума певної колонки для перевірки цілосності;

- FormatFileName – назва формат-файлу файлу, що завантажується. Містить в собі опис структури відповідного файлу, щоб база даних при завантаженні встановила відповідні зв’язки між атрибутами цільової таблиці та вхідного файлу;

- PostDate – дата отримання файлу, наприклад, -1 від поточної дати, якщо завантаження у базу даних відбувається наступного дня від отримання файлу;

- PostDate_ID – дата отримання файлу у числовому форматі, наприклад, дата 2022-01-01 у числовому представленні має значення 20220101;

- ProcessedRows – кількість реально завантажених рядків у базу даних з відповідного файлу.

Лістинг 3.8 – Код зовнішньої збірки (assembly)

```
public class LineCountProcClass
{
    [Microsoft.SqlServer.Server.SqlProcedure]
    public static void LineCount(System.String file, out int
valueVar) {
          int lineCnt = 0;
          using(var reader = new StreamReader(file))
          {
               while(reader.ReadLine() != null)
               {
                    lineCnt++;
               }
          }
          valueVar = lineCnt;
    }
}
```

У поданому коді відбувається створення нового екземпляру класу потоку для читання StreamReader для вхідного файлу, у якому за допомогою циклу відбувається підрахунок кількості рядків. Метод класу-збірки має вихідний параметр «valueVar», присвоєння значення якому відбувається після завершення циклу з підрахунку.

Для використання цього коду в SQL Server потрібно спочатку створити збірку, що буде мати посилання на відповідну бібліотеку, а також процедуру, що матиме у своїй реалізації виклик збірки (лістинг 3.9).

Лістинг 3.9 – Створення зв’язку та посилань на зовнішню бібліотеку

```
CREATE ASSEMBLY lineCounterASM
     FROM 'D:\SQL\lineCounter.dll'
     WITH PERMISSION_SET = EXTERNAL_ACCESS
GO
CREATE PROCEDURE lineCounter
     @file NVARCHAR(4000),
     @cnt INT OUTPUT
AS
EXTERNAL NAME lineCounterASM.LineCountProcClass.LineCount
GO
```

3.3.1 Стадія Extract

При розробці процедури вилучення даних в першу чергу необхідно визначити частоту вивантаження даних з OLTP-системи або окремих джерел.

Це буде займати визначений час і буде називатичя вікном вивантаження (Extract Window).

Першим кроком процесу ETL є вилучення даних (лістинг 3.10). На цьому кроці дані з різних вихідних систем витягуються у різних форматах, такі як реляційні бази даних, NoSQL, XML і плоскі файли в область для обробки (Staging Area).

Лістинг 3.10 – Частина реалізації завантаження файлу

```
DECLARE @SQLcmd VARCHAR(MAX) =
'BULK INSERT ' + @RawTableName + '
FROM ''' + @RawFilePath + @RawFileName + '''
WITH (
     FIRSTROW = ' + CONVERT(VARCHAR, @FirstRow) + ',
     LASTROW = ' + CONVERT(VARCHAR, @rowTotal) + ',
     FORMATFILE = ''' + @FormatFilePath + @FormatFileName + '''
);'
EXEC (@SQLcmd)
DECLARE @rowBulkCount INT = @@ROWCOUNT
```

У наведеному коду відбувається формування рядку для виклику виразу BULK INSERT – операції, що виконує імпорт даних у базу даних за допомогою відповідних провайдерів даних. Після виконання операції відбувається зберігання кількості завантажених рядків (@@ROWCOUNT) до попередньо оголошеної змінної @rowBulkCount, що має тип INT – вона знадобиться у майбутньому для порівняння, а також для запису до таблиці інформації про вхідні файли.

Важливо зберігати дані спочатку в області для обробки, а не безпосередньо в сховищі даних, оскільки отримані дані мають різні формати, а також можуть бути пошкоджені. Тому завантаження безпосередньо в сховище даних може пошкодити його, і відновлення буде набагато складнішим. Область для обробки дає можливість перевірити дані, яки було витягнуто, перед тим, як їх перемістити до сховища даних.

Наприклад, для датасету акаунтів та транзакцій будуть створені відповідні raw-таблиці, що наведені у лістингу 3.11.

Лістинг 3.11 – Шаблон створення raw-таблиць

```
CREATE TABLE _rawAccounts
(
     RowID INT IDENTITY(1,1) NOT NULL,
     AccountNumber VARCHAR(100) NULL,
     ...
     CONSTRAINT PK__rawAccounts PRIMARY KEY (RowID)
)
GO

CREATE TABLE _rawTXNS
(
     RowID INT IDENTITY(1,1) NOT NULL,
     AccountNumber VARCHAR(100) NULL,
     ...
     CONSTRAINT PK__rawTXNS PRIMARY KEY (RowID)
)
GO
```

Відповідно таблиці мають наступні атрибути:

- RowID – внутрішнє поле, має цілочисельний тип даних з властивістю IDENTITY, що представляє собою лічильник. Унікально ідентифікує рядки у таблиці, є первинним ключем (визначено у відповідних CONSTRAINT обмеженнях);

- AccountNumber – номер рахунку-акаунту, який унікально ідентифікує клієнта банку;

- «…» – це відповідно перелік інших вхідний колонок файлу.

На цьому етапі важливо перевірити суми рядків або певних значень (лістинги 3.12 та 3.13), бо якщо в системі більше рядків, ніж у початкових даних – це означає, що завантаження пройшло з помилкою, тому це один із найважливіших етапів процесу ETL.

Лістинг 3.12 – Перевірка кількості рядків

```
DECLARE @rowTotal INT
DECLARE @FilePath VARCHAR(8000) = @RawFilePath + @RawFileName
EXEC lineCounter @FilePath, @rowTotal OUT
...
IF (@@ROWCOUNT != @rowTotal)
     THROW 55000, 'Count of records is not matching', 1;
```

У цій ділянці коду відбувається виклик зовнішньої збірки за

допомогою процедури lineCounter, що була визначена раніше. Потім відбувається порівняння системної змінної @@ROWCOUNT зі змінною @rowTotal, що отримала своє значення від зовнішньої збірки. У випадку невідповідності викликається помилк з потрібним повідомленням.

Лістинг 3.13 – Перевірка контрольного значення

```
DECLARE @totalAmount MONEY = 0.0000
IF @AmountColumn != ''
BEGIN
     DECLARE @NSQLcmd NVARCHAR(MAX) =
        'SELECT @Amount += CAST(' + @AmountColumn + ' AS MONEY)
        FROM ' + @RawTableName + '
        WHERE Import_ID = ' + CAST(@newID AS VARCHAR)
     EXEC sp_executesql @NSQLcmd, N'@Amount MONEY OUT', @Amount
= @totalAmount OUT
END
```

Також важливо внести у таблицю, що була представлена у лістингу 3.7, інформацію про файл, що завантажується за допомогою команди INSERT (лістинг 3.14).

Лістинг 3.14 – Внесення інформації про файл у таблицю logImport

```
INSERT INTO logImport(
     ImportDateTime,
     RawTableName,
     RawFileName,
     FormatFileName,
     PostDate,
     PostDate_ID,
     ProcessedRows)
VALUES (
     GETDATE(),
     @RawTableName,
     @RawFileName,
     @FormatFileName,
     @PostDate,
     CAST(CONVERT(VARCHAR, @PostDate, 112) AS INT),
     0    )
```

Відповідно таблиця має наступні атрибути:

- GETDATE() – це, як вже було описано раніше, функція, що повертає поточне значення дати та часу;

- змінні, що починаються на «@» – це вхідні параметри до процедури

завантаження файлу;

- CONVERT(VARCHAR, @PostDate, 112) – це перетворення змінної з типом даних DATE до рядкового вигляду у форматі «YYYYMMDD» (третій параметр означає стиль, до якого має бути приведена дата, у даному випадку це 112), наприклад, дата 2022-01-01 перетвориться у рядок «20220101»;

- CAST(… AS INT) – це оператор зміни типу, у цьому випадку перетворення до числового типу INT.

Деякі додаткові кроки, що виконуються на етапі Extract:

- звірення записів з початковими даними;

- перевірка, що не завантажено спам/небажані дані;

- перевірка типів даних;

- видалення усі типів дублікатів або фрагментованих даних;

- перевірка, чи всі ключові поля є не пустими.

Вилучення даних зазвичай відбувається одним із трьох способів.

Сповіщення про оновлення: найпростіший спосіб отримати дані з вихідної системи – це сповістити цю систему, коли запис було змінено. Більшість баз даних забезпечують механізм для цього, щоб вони могли підтримувати реплікацію бази даних, і багато додатків надають веб-перехоплення, які пропонують концептуально схожу функціональність.

Поступове (інкрементальне) вилучення: деякі системи не можуть надати сповіщення про оновлення, але вони можуть визначити, які записи було змінено, і надати витяг із цих записів. Під час наступних кроків ETL система повинна ідентифікувати зміни та поширити їх.

Повне вилучення: деякі системи не можуть визначити, які дані було змінено, тому перезавантаження всіх даних є єдиним способом вивести дані з системи. Цей метод вилучення вимагає збереження копії останніх даних, щоб мати змогу перевірити, які записи є новими. Оскільки цей підхід передбачає передачу великих обсягів даних, рекомендується використовувати його лише в крайньому випадку та лише для невеликих таблиць.

3.3.2 Стадія Transform

Другим кроком процесу ETL є трансформація. На цьому етапі набір правил або функцій застосовуються до початкових даних, щоб перетворити їх у єдиний стандартний формат.

Дані, отримані з вихідного сервера, є необробленими та непридатними для використання в оригінальному вигляді. Тому його потрібно очистити від небажаних символів (лістинги 3.15 та 3.16), трансформувати (лістинг 3.18) та доповнити додатковими довідниковими даними (лістинг 3.19). Фактично, це крок, на якому процес ETL додає цінності та змінює дані, щоб можна було створювати глибокі звіти Business Intelligence. Під час цього етапу можна застосовувати правила та норми, які забезпечують якість і доступність даних. Також можна застосувати правила, щоб допомогти компанії відповідати вимогам звітності.

Для очищення вхідних даних від небажаних символів потрібно спершу визначити перелік цих символів, що повинні перевірятися і видалятися (лістинг 3.15).

Лістинг 3.15 – Створення таблиці небажаних символів

```
CREATE TABLE dicSymbols (
     ID INT IDENTITY(1,1) NOT NULL,
     symbol VARCHAR(1) NOT NULL
     )
;
INSERT INTO dicSymbols VALUES ('%');
INSERT INTO dicSymbols VALUES ('$');
...
GO
```

Таблиця має просту структуру, лише дві колонки – ID та symbol. Перша означає унікальний ідентифікатор рядку, а друга відповідно саме значення небажаного символу, які наповнюються за допомогою команд INSERT нижче, після створення таблиці.

Для самого виконання процедури очищення потрібно сформувати код, який у своїй структурі матиме оновлення кожної комірки кожного атрибуту у

вхідній таблиці: для цього потрібно використовувати динамічне формування SQL-запиту у вигляді рядку, а потім його запуск на виконання, наприклад за допомогою команди EXEC (лістинг 3.16).

Лістинг 3.16 – Процедура очищення від небажаних символів

```
CREATE OR ALTER PROCEDURE usp_CleanTable
     @TableName SYSNAME
AS
BEGIN
     DECLARE @Activity VARCHAR(1000) = OBJECT_NAME(@@PROCID)
     EXEC usp_WriteLog @Activity, 'Started'

     DECLARE @AllReplace VARCHAR(MAX) = ''
     DECLARE @AllSymbols VARCHAR(MAX) = ''
     SELECT @AllReplace += 'REPLACE(',
          @AllSymbols += ','''' + symbol + ''', '''' )'
     FROM dicSymbols

     DECLARE @SQLcmd VARCHAR(MAX) = 'UPDATE ' + @TableName + '
SET '
     SELECT @SQLcmd += (a.name + '=' + @AllReplace + a.name +
@AllSymbols + ', ')
     FROM sys.columns a JOIN sys.tables b ON a.object_id =
b.object_id
     WHERE b.name = @TableName AND a.system_type_id = 167

     SET @SQLcmd = LEFT(@SQLcmd, LEN(@SQLcmd) - 1)
     EXEC(@SQLcmd)

     EXEC usp_WriteLog @Activity, 'Succeed', @@ROWCOUNT
END
GO
```

Процедура очищення таблиці має вхідний параметр @TableName, що означає ім’я таблиці, яка повинна бути очищена. Розглянемо більш докладніше алгоритм роботи процедури.

Крок 1. Створити новий запис у таблиці логів logETL за допомогою процедури usp_WriteLog.

Крок 2. Створити та наповнити змінну @AllReplace кількістю «REPLACE(», що дорівнює кількості небажаних символів. Для інформації, команда REPLACE має три параметри:

- вхідний рядок, що потрібно змінити;

- символ для пошуку, який треба замінити (всі входження у рядок);

- символ для кінцевої заміни.

Крок 3. Створити та наповнити змінну @AllSymbols кожним символом з таблиці dicSymbols за шаблоном «, 'символ', '')».

Кроки 2 та 3 виконуються в одній конструкції SELECT для уникнення надлишкового звернення до таблиці dicSymbols та, як наслідок, покращення швидкодії.

Крок 4. Виконати конкатенацію ключових слів «UPDATE», «FROM», імені таблиці, змінної @AllReplace, назви кожної колонки вхідної таблиці та змінної @AllSymbols. Вилучення колонок відбувається з системної таблиці sys.columns, що зберігає інформацію про кожну колонку у всій базі даних, зі з'єднанням з іншою системною таблицею sys.tables, щоб обрати відповідно колонки лише для поданої таблиці. Фільтрування колонок відбувається за допомогою назви таблиці із вхідної змінної @TableName. У результаті конкатенацій формується наступний рядок, наприклад для вже визначеної таблиці dicSymbols (лістинг 3.17).

Лістинг 3.17 – Сформований скрипт для очищення таблиці dicSymbols

```
UPDATE dicSymbols SET
ID = REPLACE(REPLACE(ID, '$', ''), '%', ''),
symbols = REPLACE(REPLACE(symbols, '$', ''), '%', ''),
```

Крок 5. Видалити останній символ коми («,») зі сформованого рядка, так як після кожної колонки ставиться кома у випадку формування далі наступної та запустити на виконання цей код.

Крок 6. Виконати логування операції за допомогою виклику процедури usp_WriteLog зі статусом успішного завершення «Success».

Після завершення очищення raw-даних, можна починати наступний крок – їх трансформацію, приклад якої наведено у лістингу 3.18.

Лістинг 3.18 – Приклад трансформації для декількох полів акаунтів

```
INSERT INTO _stageAccounts (
    AccountKey,
```

```
     RegistrationDate,
     CreditAmount,
     CreditPercent,
     CreditDuration,
     ...  )
SELECT
     a.AccountNumber,
     CAST(a.RegistrationDate AS DATE),
     CAST(a.CreditAmount AS MONEY),
     CAST(a.CreditPercent AS INT),
     CAST(a.CreditDuration AS INT)
     ...
FROM _rawAccounts a
```

У цьому простому випадку відбувається перенесення даних з raw-таблиці до stage-таблиці, яка складається вже з правильних конкретних типів даних (INT, DATE, VARCHAR, тощо), а не лише рядкових, як це було в raw-таблиці, бо у випадку надходження даних з зайвими символами у числовому полі, наприклад, в тип даних INT сталась б помилка перетворення типу даних. Саме тому попередня очистка даних, як рядків, а потім їх трансформування до відповідних типів даних є доцільним рішенням.

Наступним кроком є доповнення stage-таблиць додатковими довідниковими даними, приклад якого наведено у лістингу 3.19.

Лістинг 3.19 – Приклад доповнення таблиці довідковими даними

```
UPDATE _stageAccounts SET
     City_ID = ISNULL(d1.ID, 0),
     RegistrationDate_ID = ISNULL(CAST(CONVERT(VARCHAR,
          c.RegistrationDate, 112) AS INT), 19000101),
     LastTransactionDate_ID = ISNULL(CAST(CONVERT(VARCHAR,
          c.LastTransactionDate, 112) AS INT), 19000101),
     CurrentCreditFlag_ID = CASE CurrentCreditFlag WHEN 'Y'
          THEN 1 WHEN 'N' THEN 2 ELSE 0 END,
     Age = DATEDIFF(yy, c.ClientDOB, GETDATE())
     ...
FROM _stageAccounts c
     LEFT JOIN dicCity d1 ON c.City = d1.City
     ...
```

У поданій частині коду відбувається оновлення атрибутів для датасету акаунтів, а саме:

- встановлення відповідності з довідниками шляхом проставляння ID

рядка довідника для кожного рядка stage-таблиці;

- як один із прикладів обчислень, був вирахуваний атрибут Age (Вік) – різниця між поточною датою та датою народження клієнту.

- проставлення ключів для полів, що мають тип даних дат, які потім будуть використані довідниками часу для додавання певних атрибутів.

Наочно результат доповнення даних зображено на рисунку 3.3.

На етапі Transform можна виконувати спеціальні операції з даними. Наприклад, якщо користувачу потрібна сума доходу від продажів, якої немає в базі даних. Або якщо ім'я та прізвище в таблиці знаходяться в різних стовпцях. Їх можна об'єднати перед завантаженням.

	AccountNumber	City	RegistrationDate	ClientDOB
1	UA1000000020	Novohrad-Volynskyi	2010-05-29	1989-06-29
2	UA1000000032	Donetsk	2021-12-01	1981-12-03
3	UA1000000117	Novohrad-Volynskyi	2006-03-15	1974-03-24
4	UA1000000292	Chernihiv	2013-01-09	1972-03-17
5	UA1000000373	Rome	2003-08-10	1998-01-16
6	UA1000000604	Nizhyn	2018-04-20	1999-11-10
7	UA1000000625	Kolomyia	2001-12-14	1991-09-30
8	UA1000000686	Madrid	2006-06-26	1975-03-22
9	UA1000000806	Chernihiv	2011-06-29	1999-12-30
10	UA1000000873	Naples	2002-01-10	1976-10-31

	AccountNumber	City	Oblast	Country	RegistrationDate	MonthName	DayOfWeekName	ClientDOB	Age
1	UA1000000020	Novohrad-Volynskyi	Zhytomyr	Ukraine	2010-05-29	May 2010	Saturday	1989-06-29	33
2	UA1000000032	Donetsk	Donetsk	Ukraine	2021-12-01	December 2021	Wednesday	1981-12-03	41
3	UA1000000117	Novohrad-Volynskyi	Zhytomyr	Ukraine	2006-03-15	March 2006	Wednesday	1974-03-24	48
4	UA1000000292	Chernihiv	Chernihiv	Ukraine	2013-01-09	January 2013	Wednesday	1972-03-17	50
5	UA1000000373	Rome	Rome	Italy	2003-08-10	August 2003	Sunday	1998-01-16	24
6	UA1000000604	Nizhyn	Chernihiv	Ukraine	2018-04-20	April 2018	Friday	1999-11-10	23
7	UA1000000625	Kolomyia	Ivano-Frankivsk	Ukraine	2001-12-14	December 2001	Friday	1991-09-30	31
8	UA1000000686	Madrid	Madrid	Spain	2006-06-26	June 2006	Monday	1975-03-22	47
9	UA1000000806	Chernihiv	Chernihiv	Ukraine	2011-06-29	June 2011	Wednesday	1999-12-30	23
10	UA1000000873	Naples	Naples	Italy	2002-01-10	January 2002	Thursday	1976-10-31	46

Рисунок 3.3 – Результат доповнення довідниковими даними

Перевірки та трансформації, що виконуються на етапі Transform:

- фільтрування: завантаження в сховище даних лише певних атрибутів – перерахування потрібних атрибутів у конструкції SELECT, замість використання символу «*», що означає вибір всіх атрибутів;

- очищення: заповнення значень NULL деякими значеннями за замовчуванням – наприклад, при створенні stage-таблиці додати властивість

«DEFAULT (значення)» для конкретного атрибуту;

- форматування: наприклад, об’єднання різного написання «U.S.A», «United States», та «America» в один формат «USA» – реалізація цієї трансформації може мати безліч способів, одним з яких є створення таблиці-довідника за типом «ключ-значення», де ключем є будь-яка форма написання («U.S.A», «United States», «America»), а значенням є бажане конкретне написання («USA»);

- з’єднання: об’єднання кількох атрибутів в один – звичайна конкатенація, за допомогою знаку «+» або функцій CONCAT(), CONCAT_WS(), тощо;

- розділення: розбиття одного атрибута на кілька атрибутів;

- використання правил та таблиць пошуку для стандартизації даних;

- перетворення набору символів та обробка кодування;

- перетворення одиниць вимірювання, дати, часу, валют, числове перетворення тощо;

- дедуплікація: виявлення та видалення дублікатів записів;

- реструктуризація ключів: встановлення ключових зв’язків між таблицями;

- перевірка порогового значення даних, наприклад, вік не може бути негативним;

- використання довідників для поповнення даних.

3.3.3 Стадія Load

Завантаження даних у базу даних цільового сховища даних є останнім кроком процесу ETL. У типовому сховищі даних величезний обсяг даних потрібно завантажити за відносно короткий період (наприклад, ніч). Швидкість та період завантаження залежать виключно від вимог та відрізняються від системи до системи, тому процес завантаження має бути оптимізовано для продуктивності.

У разі збою завантаження механізми відновлення повинні бути налаштовані на перезапуск із точки збою без втрати цілісності даних. Адміністратори сховища даних повинні відстежувати, відновлювати та скасовувати завантаження відповідно до переважної продуктивності сервера.

Етап завантаження процесу ETL значною мірою залежить від того, що повинно бути зроблено з даними після їх завантаження в сховище даних.

Існує два основні методи завантаження даних у сховище (таблиця 3.1).

Таблиця 3.1 – Характеристики методів завантаження даних у сховище

	Повне завантаження	Інкрементальне завантаження
Синхронізація рядків	Всі рядки з отриманих даних	Тільки нові та оновлені рядки
Час	Більше часу	Менше часу
Складність	Низька	Велика. Потрібно виявляти нові та оновлені рядки. Важче відновлення після помилки

Повне або початкове завантаження (Full Load) – у сценарії повного завантаження ETL все, що надходить із етапу перетворення, потрапляє в сховище даних.

Поступове або інкрементальне завантаження (Incremental Load) – є більш комплексним, бо порівнює вхідні дані з тими, що вже є, і створює додаткові записи, лише якщо знайдено нову унікальну інформацію.

Розглянемо у якості прикладу завантаження датасету транзакцій у кінцеве сховище даних. З боку методу завантаження припустимо, що він є повним – вхідні дані вже є профільтрованими та тільки новими на момент цього циклу завантаження (лістинг 3.20).

Лістинг 3.20 – Приклад завантаження датасету транзакцій

```
CREATE OR ALTER PROCEDURE usp_DeployTXNS
```

```
AS
BEGIN
     DECLARE @Activity VARCHAR(1000) = OBJECT_NAME(@@PROCID)
     EXEC usp_WriteLog @Activity, 'Started'
     DECLARE @AllColumns VARCHAR(MAX)
     EXEC usp_GetTableFieldsList '_stageTXNS', @AllColumns OUTPUT
     EXEC ('INSERT INTO _factTXNS (' + @AllColumns + ') SELECT ' +
@AllColumns + ' FROM _stageTXNS')
     EXEC usp_WriteLog @Activity, 'Succeed', @@ROWCOUNT
END
GO
```

Процедура починається та закінчується з уже відомого виклику логування поточного процесу. В тілі процедури виконується формування рядку з командами INSERT та SELECT – динамічним є створення переліку атрибутів за допомогою іншої процедури «usp_GetTableFieldsList», яка повертає повний список атрибутів для таблиці, назва якої передається першим аргументом. Як наслідок, це робить код більш універсальним та незалежним від наявних атрибутів та не потребує змін у тілі процедури у випадку збільшенні, зменшенні, чи перейменуванні атрибутів у stage-таблиці тощо.

3.4 Проектування моделі OLAP-компоненту

Моделювання вимірів (Dimensional Modeling, або OLAP Modeling) – це техніка для структури даних, оптимізована для зберігання у сховищі даних. Метою вимірного моделювання є оптимізація бази даних для швидшого пошуку даних. Вимірна модель у сховищі даних призначена для читання, узагальнення, аналізу числової інформації: значень, балансів, підрахунків, ваги тощо, у сховищі даних. Для протилежності, моделі відношень оптимізовані для додавання, оновлення та видалення даних у системі онлайнових транзакцій у режимі реального часу.

Ці вимірні та реляційні моделі мають свій унікальний спосіб зберігання даних, кожен з яких має певні переваги. Наприклад, для реляційної моделі нормалізація зменшує надмірність даних, а вимірна модель навпаки – упорядковує дані у сховищі даних таким чином, щоб було легше отримувати

інформацію та створювати звіти.

Елементами вимірної моделі даних є таблиці фактів та вимірів:

- факти – це метрики або «факти» з бізнес-процесу (для бізнес-процесу «Продажі» вимірюванням буде квартальне число продажів);

- вимір – надає контекст навколо події бізнес-процесу. Простіше кажучи, вони повідомляють про те, «хто», «що», «де», тощо. У бізнес-процесі «Продажі» для фактичного числа квартальних продажів будуть виміри: хто – імена клієнтів, де – розташування, що – назва продукту. Іншими словами, вимір – це розріз для перегляду інформації у фактах.

Атрибути – це різні характеристики вимірів у моделюванні. У вимірі «Розташування» атрибути можуть бути: країна, область, місто, поштовий індекс, тощо. Атрибути використовуються для пошуку, фільтрації або класифікації фактів.

Етапи OLAP моделювання:

- визначення бізнес-процесу;

- визначення «зернистості» – рівня деталізації у фактах;

- визначення вимірів;

- визначення метрик;

- побудова схеми.

На рисунку 3.4 наведено кінцеву структурну схему таблиць та їх відношень у OLAP кубі.

Крок 1: визначення фактичного бізнес-процесу, який має охоплювати сховище даних. Це може бути маркетинг, продажі, кадри тощо відповідно до потреб організації в аналізі даних. Вибір бізнес-процесу також залежить від якості даних, доступних для цього процесу. Це найважливіший етап процесу моделювання даних, і збій тут призведе до каскадних і непоправних дефектів. Щоб описати бізнес-процес, можна використовувати звичайний текст, базову нотацію моделювання бізнес-процесів (BPMN) або уніфіковану мову моделювання (UML). Для моделі, що досліджується, бізнес-процесом є банківські операції, що мають два набори даних: акаунти та транзакції.

Крок 2: визначення рівня деталізації. «Зернистість» описує рівень деталізації бізнес-проблеми/рішення. Це процес визначення найнижчого рівня інформації для будь-якої таблиці у сховищі даних. Якщо таблиця містить дані про продажі за кожен день, це має бути щоденна деталізація. Якщо таблиця містить дані про загальні продажі за кожен місяць, вона має місячну деталізацію.

Розглянемо приклад рівня деталізації: генеральний директор компанії бажає щодня відстежувати продажі певних продуктів у різних місцях. Отже, «зернистість» – це «інформація про продаж продукції за місцезнаходженням за день». Для моделі, що досліджується, рівнем деталізації є один день, адже нові дані надходять кожного дня.

Крок 3: визначення вимірів. Виміри – це такі об’єкти, як дата, магазин, місце тощо. У цих вимірах мають зберігатися всі дані. Наприклад, дата може містити такі дані, як рік, місяць і день тижня.

Приклад виміру: генеральний директор компанії бажає щодня відстежувати продажі певних продуктів у різних місцях, отже вимірами виступають: товар, місце та час. Наприклад, атрибутами для продукту є: ключ продукту (зовнішній ключ), назва, тип, специфікації. Ієрархіями для місцезнаходження є: країна, область, місто, вулиця.

Для моделі, що досліджується, будуть сформовані наступні виміри: дата відправлення транзакції, дата завершення транзакції, дата реєстрації, дата викладення файлів, дата початку «Скарбнички», дата закінчення «Скарбнички», дата останньої транзакції, дата останнього платежу за кредитом, дата початку кредиту, дата закінчення кредиту, дата закриття акаунту, різні «так або ні» виміри, тип транзакції, категорія транзакції, відділення реєстрації, працівник-регістратор, причина відмови транзакції, місце проведення транзакції (ієрархічний вимір), різні виміри «відсоток», поточний тарифний план, валюта, статус активності акаунту, різні виміри «баланс» (ієрархічний).

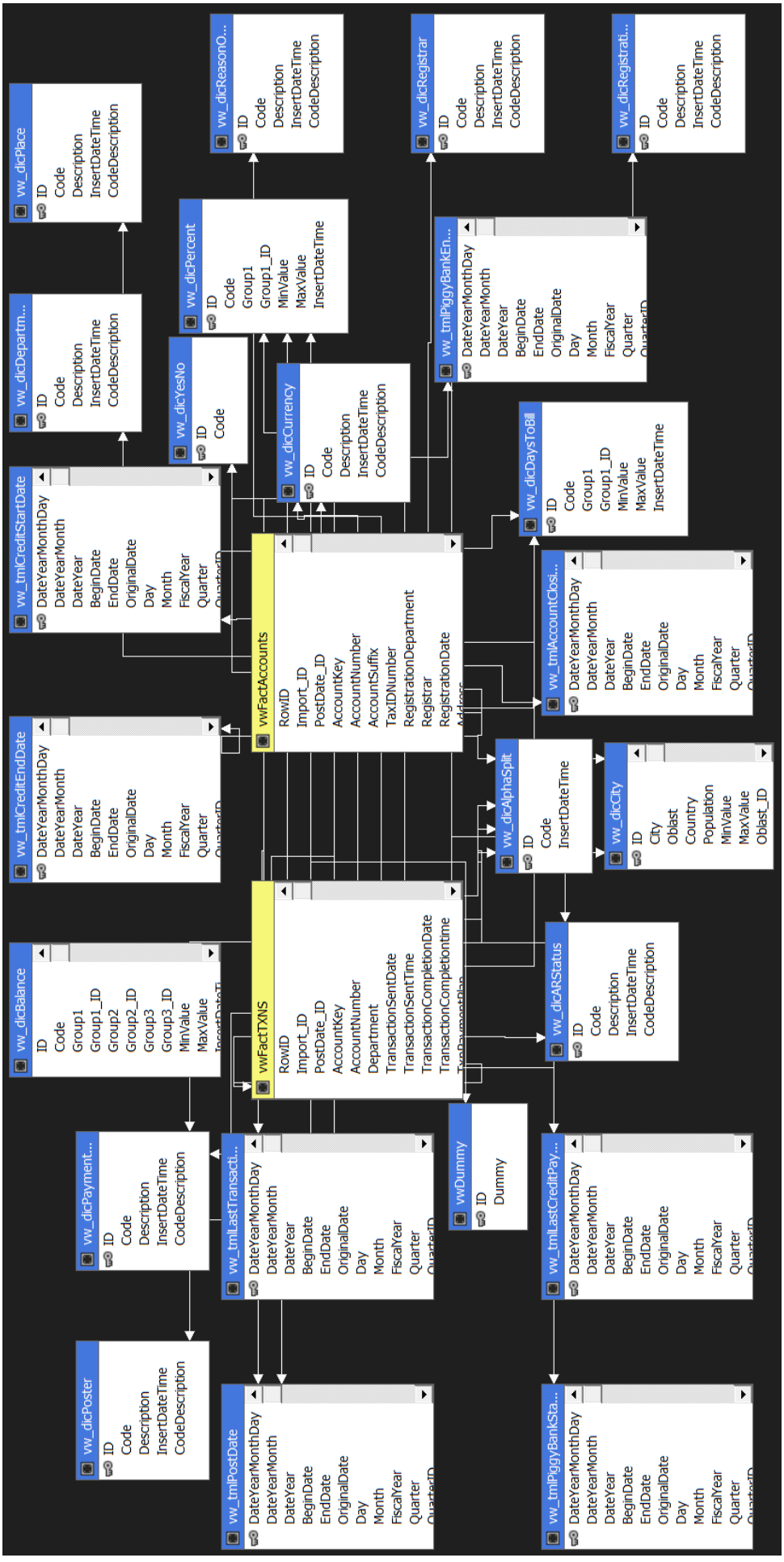


Рисунок 3.4 – Кінцева структурна схема таблиць та їх відношень у OLAP кубі

Крок 4: визначення метрик. Цей крок пов’язаний із бізнес-користувачами системи, оскільки саме тут вони отримують доступ до даних, що зберігаються в сховищі даних. Більшість рядків таблиці фактів – це числові значення, такі як ціна або вартість за одиницю тощо. Приклад визначення метрик: генеральний директор компанії бажає щодня відстежувати продажі певних продуктів у різних місцях. Метриками тут є сума продажів.

3.4.1 Проектування OLAP-вимірів

Вимір – це набір з однієї або більше рівнів ієрархій у кубі, який користувач розуміє та використовує як основу для аналізу даних. Наприклад, географічний вимір може включати рівні «Країна», «Область» та «Місто». Або вимір часу може містити ієрархію з рівнями року, кварталу, місяця та дня. У звіті або зведеній діаграмі кожна ієрархія стає набором полів, які можна розгорнути і згорнути для більш низького або вищого рівня. Кожен вимір має хоча б одну ієрархію – логічну структуру дерева, яка впорядковує члени виміру, щоб кожен з них був батьківським та мав 0 або більше членів нащадків.

Нащадок – це учасник наступного нижнього рівня ієрархії, безпосередньо пов'язаний із поточним. Наприклад, в ієрархії «Час», що містить рівні «Квартал», «Місяць» та «День», «Січень» є нащадком «Кварталу 1».

Батьківський член – це член на наступному вищому рівні ієрархії, безпосередньо пов'язаний із поточним. Батьківське значення зазвичай є консолідацією значень її дітей. Наприклад, в ієрархії часу, що містить рівні «Квартал», «Місяць» та «День», «Квартал 1» є батьківським для члену «Січень».

Виміри можна умовно поділити на наступні категорії (проте одні можуть частково належати до інших за своїм визначенням або ж до кількох одразу):

- звичайні, динамічні виміри: зазвичай це код та певні атрибути до нього, що в результаті утворюють один рівень у ієрархії;

- ієрархічні виміри: мають кілька рівнів ієрархії, не менше двох;

- часові виміри: є підвидом ієрархічних, побудовані на основі типів даних дати та/або часу, наприклад «рік – квартал – місяць – день»;

- попередньо визначені виміри: мають константну структуру та набір членів, що були визначені заповнені попередньо. Частково у своєму визначенні можуть бути віднесені до будь-яких з вищеперерахованих.

Динамічний вимір завжди є поповнюваним за своїми даними – при надходженні нових даних, він доповнюється новими унікальними значеннями ключів та атрибутів, що не існували раніше. Прикладом такого виміру є «Категорія транзакції», приклад якого наведено на рисунку 3.5.

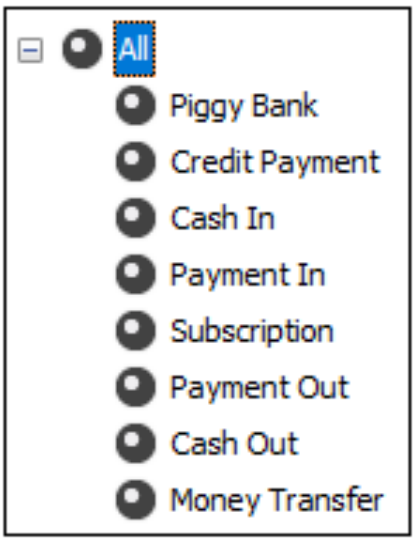
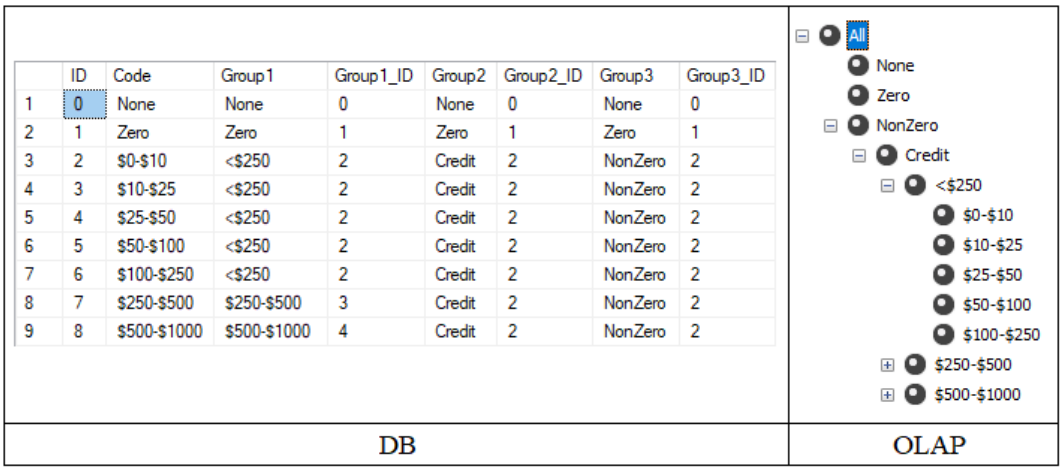


Рисунок 3.5 – Набір даних виміру «Transaction Category»

Прикладом ієрархічного виміру може бути «Balance» (Баланс). Він має 4 рівні ієрархії: «Нічого/нуль/не нуль», «Дебет/кредит», та два групових рівня абсолютних значень (рисунок 3.6). Цей вимір може бути застосований до грошових значень факт-таблиць не один раз, адже структура залишається константною, а джерело даних (колонка) може бути різною, як наслідок – будь-яка кількість таких вимірів, що побудована на одному шаблоні ієрархії, проте з різним джерелом даних.



	ID	Code	Group1	Group1_ID	Group2	Group2_ID	Group3	Group3_ID
1	0	None	None	0	None	0	None	0
2	1	Zero	Zero	1	Zero	1	Zero	1
3	2	$0-$10	<$250	2	Credit	2	NonZero	2
4	3	$10-$25	<$250	2	Credit	2	NonZero	2
5	4	$25-$50	<$250	2	Credit	2	NonZero	2
6	5	$50-$100	<$250	2	Credit	2	NonZero	2
7	6	$100-$250	<$250	2	Credit	2	NonZero	2
8	7	$250-$500	$250-$500	3	Credit	2	NonZero	2
9	8	$500-$1000	$500-$1000	4	Credit	2	NonZero	2

Рисунок 3.6 – Значення виміру «Balance» в базі даних та OLAP

Прикладом часового виміру можна розглянути «Transaction Sent Date». Він побудований для датасету транзакцій на основі однойменного поля, що означає час відправки транзакції. Має одну ієрархію та 4 її рівні: «Рік», «Квартал», «Місяць» та «День» (рисунок 3.7).

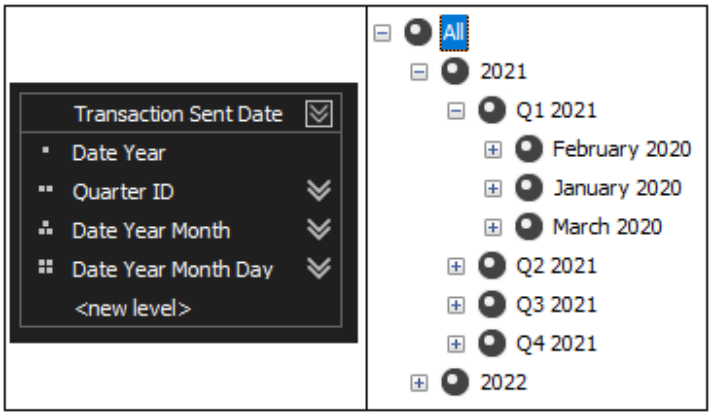


Рисунок 3.7 – Ієрархія виміру «Transaction Sent Date»

Для прикладу попередньо визначеного виміру розглянемо найпростіший екземпляр «Yes No» – має одну ієрархію, один атрибут «Код», а його члени були визначені заздалегідь у лістингу 3.21, результат якого виконання наведено на рисунку 3.8.

Лістинг 3.21 – Визначення виміру «Yes No»

```
CREATE VIEW [dbo].[vw_dicYesNo]
AS
     SELECT 0 AS ID, 'None' AS Code
```

```
      UNION ALL
      SELECT 1, 'Yes'
      UNION ALL
      SELECT 2, 'No'
GO
```

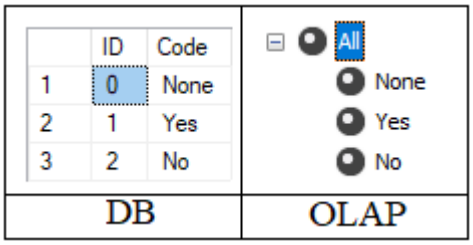


Рисунок 3.8 – Значення виміру «Yes No» в базі даних та OLAP

3.4.2 Метрики та калькуляції OLAP

Щоб розпочати моделювання метрик та калькуляцій, для початку потрібно визначити різницю між ними.

Метрика – це значення у кубі, що ґрунтується на стовпці в таблиці фактів кубу та зазвичай є числовим типом даних. Також це центральні значення в кубі, які попередньо готуються, агрегуються та аналізуються. Поширені приклади: продаж, прибуток, доходи та витрати.

Калькуляція – це значення у кубі, які обчислюються за допомогою виразу на мові MDX. Значення калькуляції можна одержати зі значень інших калькуляцій або метрик. Наприклад, «Прибуток» можна визначити шляхом віднімання значення метрики «Витрати» від значення метрики «Продажі».

Для створення метрик доступні різні типи агрегування:

- sum (сума): звичайне додавання значень;

- count or rows (кількість рядків): підрахування кількості рядків незалежно від значень;

- count of non-empty values (кількість не пустих значень): підрахування кількості рядків з не NULL значеннями;

- minimum (мінімум): мінімальне значення у вхідній колонці;

- maximum (максимум): максимальне значення у вхідній колонці;

- distinct count (відмінна кількість): кількість різних, відмінних один від

одного значень у вхідній колонці.

Розглянемо доступні поля у числовому форматі, наприклад, для датасету транзакцій:

- TxnAmount – кількість коштів, що було передано у транзакції;

- TaxAmount – абсолютне значення комісії;

- CurrentBalance – залишок на рахунку після проведення транзакції;

- CurrentCreditBalance – залишок на кредитному рахунку для аканту, який було прив’язано з відповідного акаунту;

- TaxPercent – відсоток комісії, яка була списана;

- TransactionDuration – тривалість транзакції у секундах.

Кожен ж поданих атрибутів може бути основою для побудови метрики для факт-таблиці транзакцій, приклад створення якої для поля TxnAmount наведено на рисунку 3.9.

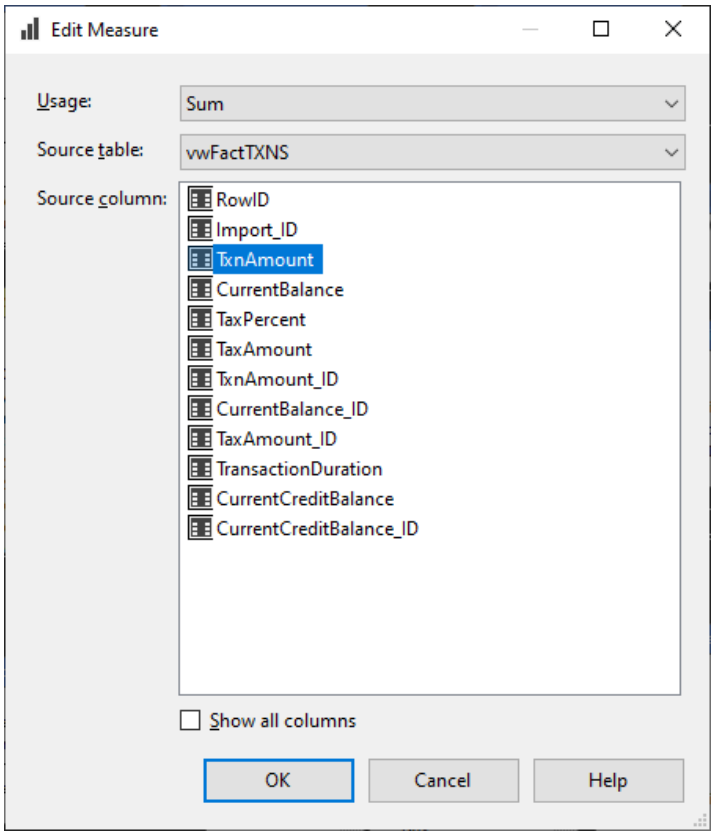


Рисунок 3.9 – Створення метрики

Калькуляція, як вже було попередньо описано, має бути написана на мові мультивимірних запитів MDX. Розглянемо створення калькуляції, що

представляє собою вирахування середнього значення розміру транзакції, а саме її головного значення TxnAmount (лістинг 3.22).

Лістинг 3.22 – Калькуляція середнього значення «TXNS Avg Txn Amount»

```
CREATE MEMBER CURRENTCUBE.[Measures].[TXNS Avg Txn Amount]
AS
    DIVIDE
    (
        [Measures].[TXNS Txn Amount],
        [Measures].[Count of TXNS]
    ),
FORMAT_STRING = "#,0.00", VISIBLE = 1, ASSOCIATED_MEASURE_GROUP
= 'TXNS';
```

У поданому лістингу використовується стандартна конструкція «CREATE MEMBER {розташування} AS», що означає початок визначення калькуляції. У тілі самої калькуляції використовується функція DIVIDE, що означає ділення та може приймати три аргументи: ділене, дільник та значення для повернення у випадку ділення на 0 (за замовчуванням = 0). У прикладі відбувається ділення метрики «Сума Txn Amount» на метрику «Кількість транзакцій», що, як наслідок, за своєю логікою означає середнє значення поля Txn Amount.

FORMAT_STRING = "#,0.00" – це формат значення, що повертається, тобто самої калькуляції в цілому. Способів визначення цієї властивості є безмежно, адже тут контролюється кількість символів для виводу всього, цілої частини, дробової частини, символ розділення тисяч, тощо. У даному випадку запис означає:

- відображати будь-яку кількість цифр у цілій частині: маска «#» означає відображати цифру або нічого (проте не менше одної), «0» означає відображати цифру або 0 у випадку її відсутності;

- відображати дві дробові цифри, із заповненням двома нулями у випадку відсутності дробових цифр – якщо значення дорівнює «1.1», то відображати «1.10», або якщо значення дорівнює «2», то відображати «2.00».

Властивість «VISIBLE» означає видимість калькуляції для сторонніх додатків (калькуляція завжди доступна у області визначення кубу) –

відповідно значення «1» означає видимість, «0» – невидимість.

ASSOCIATED_MEASURE_GROUP – це властивість, що означає відношення до якої групи метрик повинна мати калькуляція.

Провівши аналіз доступних числових колонок двох датасетів (акаунти та транзакції), а також доцільності їх використання з боку бізнес-логіки, було створено метрики, перелік яких подано на рисунку 3.10.

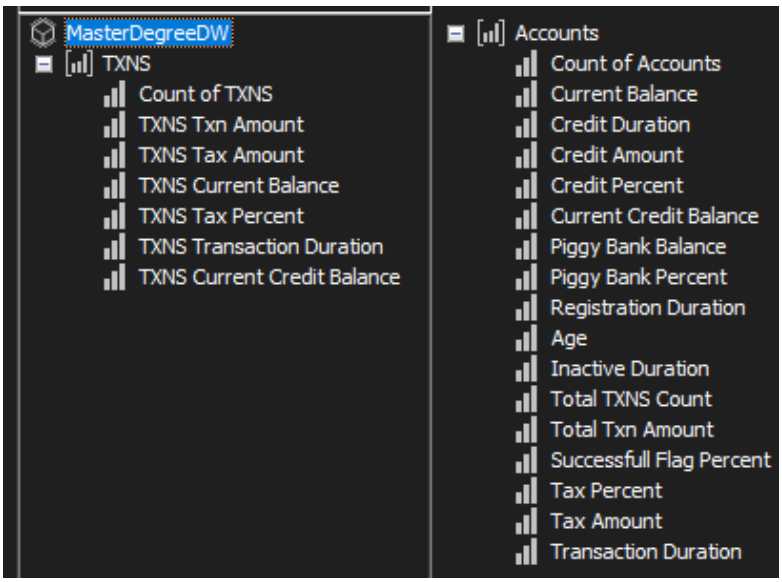


Рисунок 3.10 – Перелік створених метрик у кубі

На основі наявних метрик, інструментів та функцій, було створено наступні калькуляції, що умовно поділяються на «вираховування середнього значення» та «відсоткові значення відносно осі» (рисунок 3.11).

[TXNS Avg Txn Amount]	[Avg Registration Duration]	[TXNS Current Credit Balance as %]
[TXNS Avg Tax Amount]	[Avg Age]	[TXNS Transaction Duration as %]
[TXNS Avg Tax Percent]	[Avg Inactive Duration]	[Count of Accounts as %]
[TXNS Avg Current Balance]	[Avg Total TXNS Count]	[Current Balance as %]
[TXNS Avg Current Credit Balance]	[Avg Total Txn Amount]	[Credit Duration as %]
[TXNS Avg Transaction Duration]	[Avg Successfull Flag Percent]	[Credit Amount as %]
[Avg Current Balance]	[Avg Tax Percent]	[Current Credit Balance as %]
[Avg Credit Duration]	[Avg Tax Amount]	[Piggy Bank Balance as %]
[Avg Credit Amount]	[Avg Transaction Duration]	[Registration Duration as %]
[Avg Credit Percent]	[Count of TXNS as %]	[Inative Duration as %]
[Avg Current Credit Balance]	[TXNS Txn Amount as %]	[Total TXNS Count as %]
[Avg Piggy Bank Balance]	[TXNS Tax Amount as %]	[Total Txn Amount as %]
[Avg Piggy Bank Percent]	[TXNS Current Balance as %]	[Tax Amount as %]

Рисунок 3.11 – Перелік створених калькуляцій у кубі

4 ЕКСПЕРИМАТЕЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ МОДЕЛІ

Для виконання вимірювання ефективності розробленої моделі можна використати порівняння відносного часу виконання запитів у різних умовах:

- прості запити даних та метрик;
- запити з фільтрами;
- виміри з великою кількість членів;
- ієрархічні виміри;
- декілька вимірів, вкладених один в одного;

Microsoft SQL Server Analysis Services має певну інтеграцію з офісною програмою Microsoft Excel, що дозволяє використовувати можливості OLAP технологій за межами Visual Studio у більш зручній формі, а також без необхідності створювати окремий додаток або платформу для звернення до аналітичної системи.

4.1 Прості запити даних та метрик

Для прикладу, можна обрати 5 метрик: «Count of account» (Кількість акаунтів), «Current Balance» (Поточний баланс), «Credit Amount» (Кредитний баланс), «Avg Age» (середній вік), «Avg Total TXNS Count» (середня кількість транзакцій на акаунт).

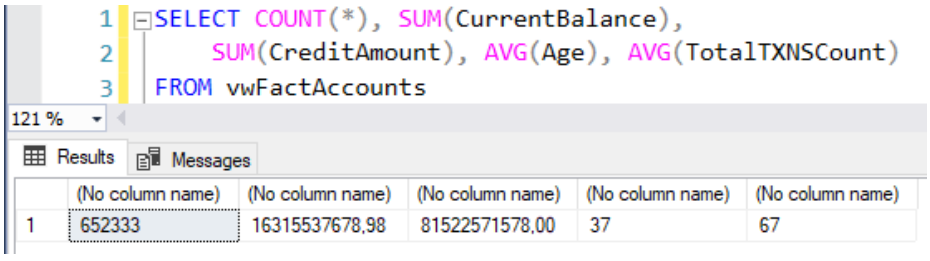


Рисунок 4.1 – Запит метрик у SQL Server

Середній час виконання у SQL Server (рисунок 4.1): 00:00.480 (хвилини, секунди та мілісекунди).

Count of Accounts	Current Balance	Credit Amount	Avg Age	Avg Total TXNS Count
652 333	16 315 537 678,98	81 522 571 578,00	37,02	67,01

Рисунок 4.2 – Запит метрик в Excel

Середній час виконання у Excel (рисунок 4.2): 00:00.465.

Як можна побачити, час виконання простих запитів є однаково незначним – це можна пояснити тим, що для кожної таблиці SQL Server збирає статистику, що робить виконання тих чи інших обчислень на повний об’єм даних у таблиці практично миттєвим.

4.2 Запити з фільтрами

Можна залишити метрики з попереднього розділу, лише додаючи декілька простих фільтрів, поля яких знаходяться у факт таблиці.

```
SELECT COUNT(*), SUM(CurrentBalance),
    SUM(CreditAmount), AVG(Age), AVG(TotalTXNSCount)
FROM vwFactAccounts
WHERE ARStatus = 'Active'
    AND Registrar = 'ID12483'
    AND MainCurrency = 'UAH'
    AND PostDate_ID = 20220630
```

21 %

Results | Messages

	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	12	272299,35	1557948,00	39	69

Рисунок 4.3 – Запит з фільтрами у SQL Server

AR Status	Active
Registrar	ID12483
Currency	UAH
Post Date	2022-06-30

Count of Accounts	Current Balance	Credit Amount	Avg Age	Avg Total TXNS Count
12	272 299,35	1 557 948,00	39,08	69,50

Рисунок 4.4 – Запит з фільтрами у Excel

Середній час виконання у SQL Server (рисунок 4.3): 00:01.800.

Середній час виконання у Excel (рисунок 4.4): 00:01.253.

Були використані подані фільтри:

- ARStatus = «Active» (статус активності акаунту);

- AccountSuffix = «Private» (суфікс акаунту);

- City = «Kyiv» (місто).

Цього разу у SQL Server спостерігається збільшення часу виконання запиту, адже окрім обчислення метрик необхідно виконати фільтрування по іншим колонкам факт-таблиці.

Наступною перевіркою є фільтрування за додатковими атрибутами, що наявні у довідникових таблицях (рисунки 4.5 та 4.6).

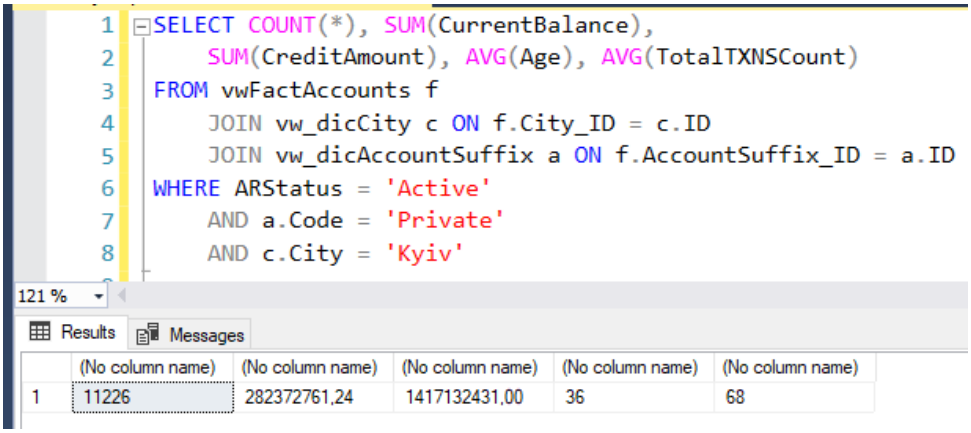


Рисунок 4.5 – Запит з фільтрами у SQL Server за додатковими атрибутами

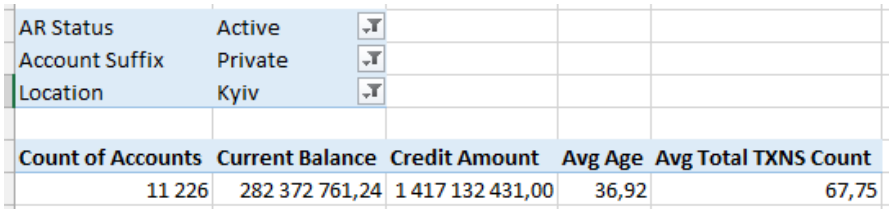


Рисунок 4.6 – Запит з фільтрами у Excel за додатковими атрибутами

Час виконання у SQL Server: 00:02.015.

Час виконання у Excel: 00:01.352.

Цього разу у обох середовищах спостерігається збільшення часу виконання запиту, адже вираховування додаткових атрибутів потребує більше ресурсів, а при їх незмінній кількості відповідно збільшується тривалість. Проте, час для OLAP практично у межах секунди, а у SQL Server виріс до п’яти секунд.

4.3 Виміри з великою кількість членів

Для виконання перевірки цього випадку потрібно обрати вимір, що містить велику кількість членів у своїй структурі, зазвичай це автопоповнювані довідники. Для прикладу, вимір Registrar – код співробітника, що виконував реєстрації клієнта. У даному випадку цей довідник містить 5048 унікальних рядків.

```
SELECT r.Code, COUNT(*), SUM(CurrentBalance),
    SUM(CreditAmount), AVG(Age), AVG(TotalTXNSCount)
FROM vwFactAccounts f
    JOIN vw_dicRegistrar r ON f.Registrar_ID = r.ID
GROUP BY r.Code
ORDER BY r.Code
```

121 %

Results | Messages

	Code	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	ID10000	143	3747896,11	18947123,00	38	67
2	ID10001	131	3248713,20	15372090,00	37	68
3	ID10002	135	3083137,74	13595762,00	38	69
4	ID10003	137	3479262,08	16579192,00	37	65
5	ID10004	149	3627104,89	16959329,00	36	67
6	ID10005	142	3860144,91	19780374,00	37	66
7	ID10006	113	3130713,38	11357332,00	37	69
8	ID10007	126	3198371,00	18682567,00	37	68
9	ID10008	137	3224822,31	17629602,00	36	71

Рисунок 4.7 – Запит виміру з великою кількість членів у SQL Server

Названия строк	Count of Accounts	Current Balance	Credit Amount	Avg Age	Avg Total TXNS Count
ID10001	131	3 248 713,20	15 372 090,00	37,19	68,51
ID10002	135	3 083 137,74	13 595 762,00	38,65	68,42
ID10005	142	3 860 144,91	19 780 374,00	37,72	66,66
ID10007	126	3 198 371,00	18 682 567,00	37,36	68,09
ID10009	126	3 285 557,31	17 063 995,00	36,46	64,97
ID10010	135	3 288 443,44	19 021 764,00	37,27	69,79
ID10016	148	3 611 276,58	19 351 695,00	37,78	70,39

Рисунок 4.8 – Запит виміру з великою кількість членів у Excel

Час виконання у SQL Server (рисунок 4.7): 00:02.202.

Час виконання у Excel (рисунок 4.8): 00:01.468.

Як можна побачити, аналітична OLAP система все ще тримається на позначці в одну секунду, при тому, що, можливо, сповільнення може бути пов’язане з роботою і отриманням даних зі сторони Excel, а не з продуктивністю OLAP.

4.4 Ієрархічні виміри

Прикладом ієрархнічного виміру є довідник Balance, що може бути використаним для будь-якого поля, що має дробовий тип (з точки зору бізнес-логіки – грошові поля). Він має 4 рівня ієрерхії (в запиті це Group1, Group2, Group3 та Code).

У даному випадку було використане поле CurrentBalance, що означає поточний баланс на рахунку акаунту. На рисунку 4.9 наведений запит у SQL Server, а на рисунку 4.10 – запит у Excel.

```
SELECT b.Group1, b.Group2, b.Group3, b.Code,
    COUNT(*), SUM(CurrentBalance),
    SUM(CreditAmount), AVG(Age), AVG(TotalTXNSCount)
FROM vwFactAccounts f
    JOIN vw_dicBalance b on f.CurrentBalance_ID = b.ID
GROUP BY b.ID, b.Group1, b.Group2, b.Group3, b.Code
ORDER BY b.ID
```

121 %

Results | Messages

	Group1	Group2	Group3	Code	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	<$250	Debit	NonZero	$10-$25	240	4099,80	28763398,00	36	66
2	<$250	Debit	NonZero	$25-$50	323	12236,65	41037240,00	36	68
3	<$250	Debit	NonZero	$50-$100	629	47666,19	74949272,00	37	69
4	<$250	Debit	NonZero	$100-$250	1953	341200,33	244580719,00	37	69
5	$250-$500	Debit	NonZero	$250-$500	3293	1229994,76	424172630,00	37	67
6	$500-$1000	Debit	NonZero	$500-$1000	6533	4907162,31	788546662,00	36	68
7	$1000-$2500	Debit	NonZero	$1000-$2500	19791	34641262,73	2463346960,00	37	67
8	$2500-$5000	Debit	NonZero	$2500-$5000	32505	121929071,96	4056126595,00	37	67
9	$5000-$10000	Debit	NonZero	$5000-$10000	65459	490894951,08	8155087440,00	37	67
10	$10000-$25000	Debit	NonZero	$10000-$25000	195081	3412185228,18	24385589990,00	37	67
11	$25000-$50000	Debit	NonZero	$25000-$50000	326515	12248794797,69	40858569106,00	37	67
12	$50000-$100000	Debit	NonZero	$50000-$100000	11	550007,30	1801566,00	36	79

Рисунок 4.9 – Запит ієрархічного виміру в SQL Server

Названия строк	Count of Accounts	Current Balance	Credit Amount	Avg Age	Avg Total TXNS Count
NonZero	**652 333**	**16 315 537 678,98**	**81 522 571 578,00**	**37,02**	**67,01**
Debit	**652 333**	**16 315 537 678,98**	**81 522 571 578,00**	**37,02**	**67,01**
<$250	3 145	405 202,97	389 330 629,00	37,06	68,13
$10-$25	240	4 099,80	28 763 398,00	36,68	65,74
$25-$50	323	12 236,65	41 037 240,00	36,99	67,67
$50-$100	629	47 666,19	74 949 272,00	37,03	68,51
$100-$250	1 953	341 200,33	244 580 719,00	37,13	68,37
$250-$500	3 293	1 229 994,76	424 172 630,00	37,14	66,46
$500-$1000	6 533	4 907 162,31	788 546 662,00	36,74	68,21
$1000-$2500	19 791	34 641 262,73	2 463 346 960,00	37,08	66,94
$2500-$5000	32 505	121 929 071,96	4 056 126 595,00	37,11	66,88
$5000-$10000	65 459	490 894 951,08	8 155 087 440,00	37,04	66,96
$10000-$25000	195 081	3 412 185 228,18	24 385 589 990,00	37,02	67,05
$25000-$50000	326 515	12 248 794 797,69	40 858 569 106,00	37,01	66,98
$50000-$100000	11	550 007,30	1 801 566,00	36,73	79,55
Общий итог	**652 333**	**16 315 537 678,98**	**81 522 571 578,00**	**37,02**	**67,01**

Рисунок 4.10 – Запит ієрархічного виміру в Excel

Час виконання у SQL Server (рисунок 4.9): 00:03.120.

Час виконання у Excel (рисунок 4.10): 00:02.241.

Цього разу SQL Server є повільнішим через те, що довелося виконувати групування по чотирьом полям, адже вимір є ієрархічним з чотирьма рівнями, тому, щоб відтворити його структуру у SQL-запиті, було потрібно використати групування.

4.5 Декілька вимірів, вкладених один в одного

Вкладені вимірі один в одного мають назву «nested dimensions». Цей спосіб використання аналітичної системи є найбільш ресурсозатратним, особливо при використанні вимірів, що містять багато членів у своїй структурі. Наприклад, один член другого рівня вимірів може відноситись до кількох членів первого рівня вимірів, і, як наслідок, таким чином потрібне вираховування декартового добутку двох рівнів у таких випадках.

При цьому, збільшення кількості рівнів прямопропорційно впливає на продуктивність системи, адже вона зменшується більше, ніж в арифметичній прогресії, можливо, навіть в геометричній в залежності від кількості членів у вимірах та їх складності.

```
1  SELECT t.DateYear, t.DateYearMonth, t.DateYearMonthDay,
2      c.Code, a.Code,
3      COUNT(*), SUM(CurrentBalance),
4      SUM(CreditAmount), AVG(Age), AVG(TotalTXNSCount)
5  FROM vwFactAccounts f
6      JOIN vw_tmlPostDate t ON f.PostDate_ID = t.DateYearMonthDay
7      JOIN vw_dicCurrency c ON f.MainCurrency_ID = c.ID
8      JOIN vw_dicARStatus a ON f.ARStatus_ID = a.ID
9  GROUP BY t.DateYear, t.DateYearMonth, t.DateYearMonthDay,
10     c.Code, a.Code
11 ORDER BY t.DateYear, t.DateYearMonth, t.DateYearMonthDay,
12     c.Code, a.Code
```

121 %

Results | Messages

	DateYear	DateYearMonth	DateYearMonthDay	Code	Code	(No column name)	(No column name)	(No column name)	(No column name)	(No column name)
1	2022	202201	20220131	EUR	Active	487	12211085,93	58585983,00	36	19
2	2022	202201	20220131	EUR	Inactive	5	128291,36	448100,00	31	19
3	2022	202201	20220131	PLN	Active	483	11837451,62	58329361,00	36	19
4	2022	202201	20220131	PLN	Inactive	5	126143,95	1110397,00	31	19
5	2022	202201	20220131	UAH	Active	48018	1200154815,38	6001251789,00	36	18
6	2022	202201	20220131	UAH	Inactive	468	11901023,59	51010449,00	37	19

Рисунок 4.11 – Запит вимірів вкладених один в одного в SQL Server

Названия строк	Count of Accounts	Current Balance	Credit Amount	Avg Age	Avg Total TXNS Count
2023	**652 333**	**16 315 537 678,98**	**81 522 571 578,00**	**37,02**	**67,01**
PLN	**6 465**	**161 989 649,08**	**817 430 512,00**	**37,16**	**67,51**
Active	6 405	160 341 904,88	809 419 505,00	37,16	67,47
Inactive	60	1 647 744,20	8 011 007,00	37,75	71,50
UAH	**632 942**	**15 828 995 047,85**	**79 105 303 870,00**	**37,02**	**67,02**
Active	626 676	15 671 792 137,55	78 333 507 631,00	37,02	67,01
Inactive	6 266	157 202 910,30	771 796 239,00	36,92	67,63
EUR	**6 476**	**163 310 140,93**	**792 319 928,00**	**36,85**	**66,64**
Active	6 416	161 778 868,17	783 814 179,00	36,83	66,63
Inactive	60	1 531 272,76	8 505 749,00	38,60	67,67
USD	**6 450**	**161 242 841,12**	**807 517 268,00**	**36,88**	**65,92**
Active	6 389	159 579 531,62	801 357 599,00	36,87	65,89
Inactive	61	1 663 309,50	6 159 669,00	38,70	68,82
Общий итог	**652 333**	**16 315 537 678,98**	**81 522 571 578,00**	**37,02**	**67,01**

Рисунок 4.12 – Запит вимірів вкладених один в одного в Excel

Час виконання у SQL Server (рисунок 4.11): 00:04.220.

Час виконання у Excel (рисунок 4.12): 00:03.123.

Як можна побачити, у цьому випадку час виконання суттєво збільшився у обох системах – як вже було описано вище, вкладені вимірі потребують значних ресурсів для обчислення, особливо якщо їх складність та кількість вкладеності збільшується.

4.6 Результати експериментів

За результатами проведених експериментів було побудовано порівняльну діаграму, що зображена на рисунку 4.13, легенду якої у свою чергу наведено таблиці 4.1.

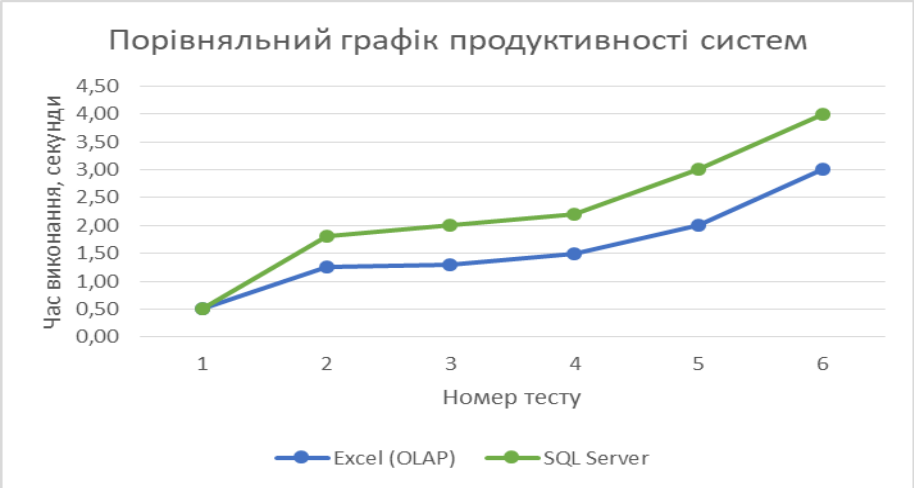


Рисунок 4.13 – Порівняльний графік продуктивності систем

Таблиця 4.1 – Легенда тестів

Номер тесту	Вид тесту
1	Запит метрик
2	Запит з фільтрами
3	Запит з фільтрами за додатковими атрибутами
4	Запит виміру з великою кількістю членів
5	Запит ієрархічного виміру
6	Запит вкладених вимірів

Можна зробити висновок, що в залежності від ресурсозатратності обидві системи мають різне збільшення часу виконання, проте OLAP система має суттєво нижчий час у будь-якому з експериментів до 25% в залежності від комплексності вхідного запиту, що робить таку систему ефективною у використанні.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було розроблено модель OLAP-компоненту та бази даних, що виступає у ролі обробника, підготовки та трансформації отриманих вхідних даних банківських операцій, а також є основою та джерелом для подальшої побудови на ній моделі для аналітичних цілей за допомогою можливостей Business Intelligence технологій.

Проведений аналіз існуючих на даний час систем аналітики дозволив визначити та сформувати вимоги і критерії, яким має відповідати аналітична Business Intelligence модель, що проектується:

- наявність повного ETL-процесу, що включає в себе завантаження файлів, їх перетворення та приведення до відповідної нормальної форми, з наступним завантаженням їх у сховище даних;
- завантаження файлів із різних джерел та різного формату;
- валідація вхідних наборів;
- обробка та очищення вхідного набору даних на можливі помилкові символи у значеннях;
- можливість використання додаткових довідників, які можуть формуватися на клієнтській стороні, що потім будуть використані для встановлення відповідності описів та інших додаткових атрибутів для відображення цих даних у таблиці фактів, а також у OLAP аналітичній формі даних;
- логування ETL-процесу для його майбутнього відстеження з метою оптимізації проблемних процедур або функцій, що можуть займати багато часу і можуть бути використані більш ефективно;
- синхронізація з історичними чи минулими даними задля можливості простеження тих чи інших тенденцій, проставлення відповідних міток для рядків даних;
- синхронізація двох датасетів між собою – з’єднання та передача калькульованих полів, щоб мати можливість швидко аналізувати.

При формуванні потреб до OLAP-компоненту також був проведений аналіз бізнес-логіки банківських операцій, завдяки чому було створено актуальні та необхідні інструменти для системи аналізу даних, а саме виміри, метрики, калькуляції, тощо.

Також, однією з переваг використання аналітичної OLAP-системи є відсутність необхідності знати мову запитів SQL, адже як було вказано вище, для неї присутня безмежна інтеграція у різні додатки, навіть у офісну програму Excel. Це додає ефективності для аналізу даних, тому що не буде необхідності, для прикладу, витрачати час та ресурси на вивчення мови запитів SQL.

При розробці архітектури як в базі даних, так і в OLAP-компоненті, було виконано структуризацію та логічне розділення об’єктів, що належать до різних датасетів і мають різне бізнес-значення. Як наслідок, наприклад, процедури у базі даних є логічно розділеними та незалежними від кількості наборів вхідних даних, що означає високу гнучкість процесу та більш просте внесення змін у випадку необхідності.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Krzysztof J. Cios. Data Mining: A Knowledge Discovery Approach. Springer, 2007. 621 p.

2. Claude Seidman. Data Mining with Microsoft SQL Server 2000 Technical Reference. Microsoft Press, 2001. 400 p.

3. Kathleen B Hass, Richard Vander Horst, Kimi Ziemski. From Analyst to Leader: Elevating the Role of the Business Analyst Management Concepts. Management Concepts Press, 2008. 152 p.

4. Teo Lachev. Applied Microsoft Analysis Services 2005 and Microsoft Business Intelligence Platform. Prologika Press, 2005. 712 p.

5. К. Дж. Дейт. Введение в системы баз данных. М.: Вильямс, 2005. 1328 с.

6. Роберт Виейра. Основные сведения о языке T-SQL. Программирование баз данных MS SQL Server 2005 для профессионалов. М.: «Диалектика», 2007. 1072 с.

7. Роберт Э. Уолтерс, Майкл Коулс. SQL Server 2008: ускоренный курс для профессионалов. М.: «Вильямс», 2008. 768 с.

8. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Часть II. Язык программирования Transact-SQL. Освоение Microsoft SQL Server 2005. М.: «Диалектика», 2007. 280 с.

9. Майк Гандерлой, Джозеф Джорден, Дейвид Чанц. Освоение Microsoft SQL Server 2005. М.: «Диалектика», 2007. 1104 с.

10. George Spofford. MDX Solutions: With Microsoft SQL Server Analysis Services. Wiley, 2001. 416 p.

11. Сивакумар Харинатх, Мэтт Кэррол. Microsoft SQL Server Analysis Services 2008 и MDX для профессионалов. М.: «Диалектика», 2010. 1072 с.

12. Sivakumar Harinath, Stephen Quinn. Professional SQL Server Analysis Services 2005 with MDX. Wrox, 2006. 864 p.

13. Reed Jacobson. Microsoft SQL Server 2000 Analysis Services Step by

Step. Microsoft Press, 2000. 400 p.

14. Mosha Pasumansky, Mark Whitehorn, Robert Zare. Fast Track to MDX. Springer, 2007. 349 p.

15. ZhaoHui Tang, Jamie MacLennan. Data Mining with SQL Server 2005. Wiley, 2005. 480 p.

16. Edward Melomed, Irina Gorbach, Alexander Berger, Py Bateman. Microsoft SQL Server 2005 Analysis Services. Sams, 2006. 842 p.

17. David Loshin. ETL (Extract, Transform, Load). Business Intelligence. 2nd. Morgan Kaufmann, 2012. 400 p.

18. David Haertzen. ETL Tools. The Analytical Puzzle: Profitable Data Warehousing, Business Intelligence and Analytic. Technics Publications, 2012. 346 p.

19. Ralph Kimball, Joe Caserta. The Data Warehouse ETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data. John Wiley & Sons, 2004. 528 p.

20. Bruno Oliveira, Óscar Oliveira, Telmo Matos, Vasco Santos, Orlando Belo. An ETL pattern for log configuration and analysis. IADIS International Conference Big Data Analytics, Data Mining and Computational Intelligence 2019 (part of MCCSIS 2019). edit.: Ajith P. Abraham, Jörg Roth, 2019. 39-46 p. ISBN 978-989-8533-92-0.

21. Офіційна документація Microsoft. URL: https://learn.microsoft.com.

22. Філімончук Т.В., Мартовицький В.О., Лифар Д.С. Модель OLAP-гіперкуба системи аналізу банківських операцій. Проблеми інформатизації: Тези доповідей десятої міжнародної науково-технічної конференції. Т.2: секція 4. Черкаси: ЧДТУ; Баку: ВАЗС АР; Бельсько-Бяла: УТіГН; Харків: НТУ «ХПІ»;. Харків: ХНУРЕ; Харків: ДП «ПД ПКНДІ АП». 2022. с. 64.