Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____Дослідження процесних моделей міграції даних в ІС_____
_____електронної_комерції_____
_____
(тема)

Виконав:
здобувач _2_ року навчання,
групи _ІУСТм-23-1____
_Митченко Ілля Володимирович_____
(прізвище, ім'я, по батькові)
Спеціальність _122 Комп'ютерні науки___
(код і повна назва спеціальності)
Тип програми ____освітньо-професійна___
(освітньо-професійна або освітньо-наукова)
Освітня програма _Інформаційні_____
управляючі системи та технології_____
(повна назва освітньої програми)
Керівник_____Сергій ЧАЛИЙ_____
(Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ ___Костянтин ПЕТРОВ__
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Інформаційних управляючих систем

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

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

Освітня програма Інформаційні управляючі системи та технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

« 09 » грудня 20 24 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Митченку Іллі Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження процесних моделей міграції даних в ІС електронної комерції

затверджена наказом університету від 27 листопада 2024 р. № 1249Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 19 січня 2025 р.

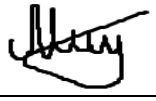
3. Вихідні дані до роботи відомості про загальний процес роботи ІС електронної комерції, відомості про узагальнений процес міграції даних, відомості про процесні моделі міграції даних в ІС електронної комерції

4. Перелік питань, що потрібно опрацювати в роботі аналіз процесних моделей міграції даних та постановка задачі дослідження; інформаційна технологія міграції даних в системах електронної комерції; практична реалізація та експериментальна перевірка процесної моделі міграції даних в ІС електронної комерції

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Видача завдання на кваліфікаційну роботу	09.12.2024	Виконано
2	Пошук необхідної літератури	09.12.2024 – 10.12.2024	Виконано
3	Опис та аналіз структурних і функціональних особливостей предметної області	10.12.2024 – 12.12.2024	Виконано
4	Огляд та аналіз сучасного стану розглянутої проблеми, а також існуючих методів і засобів вирішення завдань кваліфікаційної роботи	12.12.2024 – 15.12.2024	Виконано
5	Формулювання задачі дослідження	16.12.2024	Виконано
6	Опис архітектури об’єкта дослідження на рівні функцій	16.12.2024 – 17.12.2024	Виконано
7	Опис удосконаленого об’єкта дослідження	17.12.2024 – 21.12.2024	Виконано
8	Розробка удосконаленого об’єкта дослідження	21.12.2024 – 09.01.2025	Виконано
9	Оформлення пояснювальної записки	09.01.2025 – 14.01.2025	Виконано
10	Перевірка на плагіат	15.01.2025	Виконано
11	Попередній захист кваліфікаційної роботи	20.01.2025	Виконано
12	Захист кваліфікаційної роботи	22.01.2025	Виконано

Дата видачі завдання __09___ ____грудня___2024 р.

Здобувач _____  _____
(підпис)

Керівник роботи 
(підпис)

_проф. каф. ІУС, Сергій ЧАЛИЙ__
(посада, власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота містить 80 с., 28 рис., 10 табл., 21 джерело.

ВЕЛИКІ МОВНІ МОДЕЛІ, ВИТЯГ, ЗАВАНТАЖЕННЯ, МІГРАЦІЇ ДАНИХ, ПЛАН МІГРАЦІЇ, РЕЛЯЦІЙНА СУБД, РЕПЛІКАЦІЯ, ТРАНСФОРМАЦІЯ, ETL, SQL.

Об’єктом дослідження виступає процес міграції даних в інформаційних системах електронної комерції.

Метою даної роботи є дослідження та удосконалення процесів міграції даних для підвищення ефективності інформаційних систем електронної комерції на основі використання сучасних інструментів, запитів до великих мовних моделей та використання процесних моделей.

Дослідження, що будуть проведені, базуються на результатах аналізу існуючих підходів до міграцій даних, можливості їх застосування між різними системами управління базами даних.

У результаті роботи були досліджені проблеми у існуючих методах міграцій, запропоновано методи їх вирішення, проаналізовано процес планування міграцій, сформульовані загальні рекомендації та спроектовано базу даних для проведення досліджень.

ABSTRACT

Explanatory note to the qualification work: 80 p., 10 tables, 28 images, 21 sources.

DATA MIGRATIONS, ETL, EXTRACTION, LARGE LANGUAGE MODELS, LOADING, MIGRATION PLAN, RELATIONAL DATABASE, REPLICATION, TRANSFORMATION, SQL.

The object of research of the qualification work is the process of data migration in e-commerce information systems.

The purpose of this work is to research and improve data migration processes to improve the efficiency of e-commerce information systems based on the use of modern tools, queries to large language models and the use of process models.

The studies that will be conducted are based on the results of the analysis of existing approaches to data migrations, the possibility of their application between different database management systems.

As a result of the work, problems in existing migration methods were investigated, methods for solving them were proposed, the migration planning process was analyzed, general recommendations were formulated and a database for research was designed.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧКИ

ІС – інформаційна система

ЕК – електронна комерція

СУБД – система управління базами даних

ETL – Extract, Transform, Load

JSON – Javascript object notation

LLM – Large language model

PaaS – Platform as a service

SQL – Structured query language

ВСТУП

Сучасне суспільство інтегрувало використання комп'ютерів майже у всі аспекти життєдіяльності. Інтенсивний розвиток інформаційних технологій супроводжується експоненціальним зростанням обсягів даних, що потребує створення і вдосконалення спеціалізованих інструментів для їх обробки та зберігання. У кожній галузі існують унікальні варіанти програмного забезпечення, кожне з яких використовує індивідуальні сховища даних, представлені здебільшого реляційними системами управління базами даних (СУБД). Ці системи постійно вдосконалюються, з'являються нові сервіси та програмні комплекси, спрямовані на підвищення ефективності роботи з даними.

Метою роботи є дослідження та удосконалення процесних моделей міграції даних шляхом узгодження даних з використання великих мовних моделей для підвищення ефективності інформаційних систем електронної комерції.

На сьогодні реляційна модель даних є домінуючим підходом до створення баз даних. Це пояснюється її інтуїтивно зрозумілою структурою, де інформація організована у вигляді таблиць. Така зручність сприяла появі численних реляційних СУБД із різними характеристиками, що робить вибір оптимальної системи управління даними складним завданням.

Із часом обсяги та кількість даних постійно зростають, що стимулює вдосконалення й модернізацію існуючих сховищ. Хоча може здаватися, що проблеми виникають лише на етапі початкового проєктування баз даних, на практиці значно складнішою задачею є підтримка та еволюція таких сховищ. Цей процес часто супроводжується необхідністю повної міграції програмної системи на оновлену версію сховища або навіть перенесення

даних на абсолютно іншу платформу. Саме це і визначає актуальність завдань, пов’язаних із міграцією даних.

У зв’язку з тим фактом, що більшість існуючих процесних моделей міграції даних основані на ручному співставленні таблиць та атрибутів, створенні проміжних програмних модулів для коректності роботи системи, можна сказати, що це забирає досить велику частку часу під час реалізації проектів з міграційних рішень. Існує велика частина етапів, які потребують автоматизації та спрощення.

1 АНАЛІЗ ПРОЦЕСНИХ МОДЕЛЕЙ МІГРАЦІЇ ДАНИХ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз особливостей ІС електронної комерції

Інформаційні системи електронної комерції є ключовим елементом сучасного бізнесу, який функціонує в цифровому середовищі. Вони забезпечують автоматизацію процесів купівлі-продажу, управління даними та взаємодії між учасниками ринку. Основними складовими таких систем є веб-додатки, бази даних, інтерфейси для користувачів, а також технології обробки та захисту інформації.

До основних особливостей ІС електронної комерції належать:

– цифрова інтеграція процесів;
– підтримка масштабованості;
– забезпечення безпеки даних;
– підтримка багатоканальності;
– аналіз даних та персоналізація;
– гнучкість інтеграції.

Цифрова інтеграція процесів полягає в тому, що системи інтегрують бізнес-процеси, включаючи управління замовленнями, обробку платежів, логістику та підтримку клієнтів, забезпечуючи цілісний підхід до ведення бізнесу.

Завдяки підтримці масштабованості і адаптивній архітектурі, ІС можуть ефективно обробляти великі обсяги даних та масштабуватись відповідно до потреб бізнесу, зокрема під час сезонних коливань попиту.

Забезпечення безпеки даних полягає в тому, що електронна комерція тісно пов’язана з обробкою конфіденційної інформації, тому ІС включають розширені механізми захисту даних, такі як шифрування, багаторівнева автентифікація та резервне копіювання.

Шляхом підтримки багатоканальності сучасні системи дозволяють вести бізнес через різні канали: веб-сайти, мобільні додатки, соціальні мережі та інші платформи.

Аналіз даних та персоналізація полягає в тому, що інформаційні системи використовують методи аналізу великих даних (Big Data) для прогнозування попиту, оптимізації маркетингових стратегій та персоналізації взаємодії з клієнтами.

Гнучкість інтеграції дозволяє ІС інтегруватися з іншими корпоративними системами, такими як CRM (системи управління взаємовідносинами з клієнтами), ERP (системи управління ресурсами підприємства) та інші.

Загальна схема роботи ІС електронної комерції наведена на рисунку 1.1.

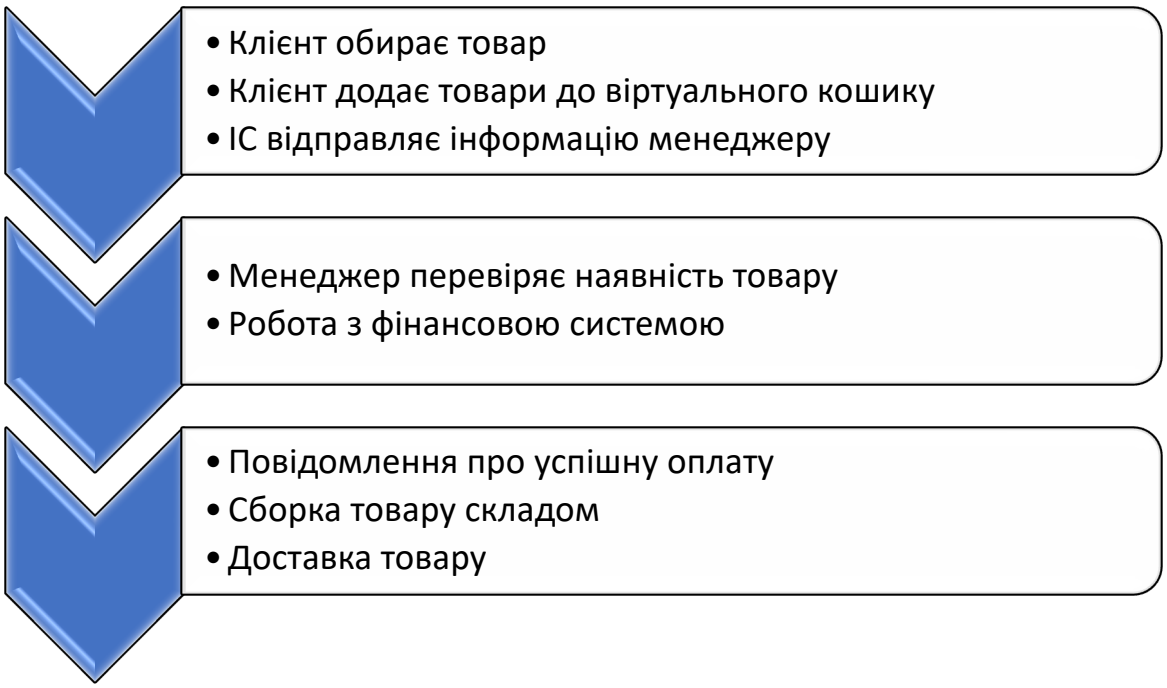


Рисунок 1.1 – Загальна схема роботи ІС ЕК

Розвиток інформаційних систем у сфері електронної комерції тісно пов’язаний з інноваціями у сфері інформаційних технологій. Важливим

напрямом є використання хмарних обчислень, штучного інтелекту та блокчейн-технологій, що дозволяють підвищити ефективність процесів і забезпечити нові можливості для бізнесу. Саме аналіз є критично важливим для розуміння механізмів їхньої роботи, визначення потенційних ризиків та можливостей удосконалення, що є основою для подальшого дослідження процесів міграції подібних систем.

Також при аналізі особливостей інформаційних систем електронної комерції важливо звернути увагу на ключові відмінності між традиційною архітектурою систем, яка може бути дуже різною. Наприклад, ІС що працює на власному сервері замовника (on-premise), та за допомогою сучасних хмарних рішень. У цьому контексті розглянемо дві версії інформаційної системи електронної комерції: стару версію на базі Hybris 6.0 та нову хмарну платформу SAP Commerce 2211.

Стара версія системи (Hybris 6.0, on-premise):

– архітектура: система працює на серверах замовника, що потребує значних ресурсів для їхньої підтримки та адміністрування. Відповідальність за оновлення, безпеку та резервне копіювання лежить на компанії;

– база даних: таблиці бази даних мають спрощену структуру зі слабкими зв’язками, що обмежує можливості для аналізу даних і гнучкого масштабування, відсутні аналітичні атрибути, старі формати індексів, які не дають максимального очікуваного результату;

– продуктивність: продуктивність системи залежить від потужності локального обладнання, яке може швидко застаріти;

– можливості оновлення: процес оновлення системи складний і пов’язаний з високими витратами, оскільки кожне оновлення потребує значних зусиль від ІТ-команди;

– безпека: безпека залежить від налаштувань компанії, що підвищує ризик виникнення вразливостей через людський фактор або недостатнє оновлення захисних механізмів.

На рисунку 1.2 зображено схему ІС старої версії.

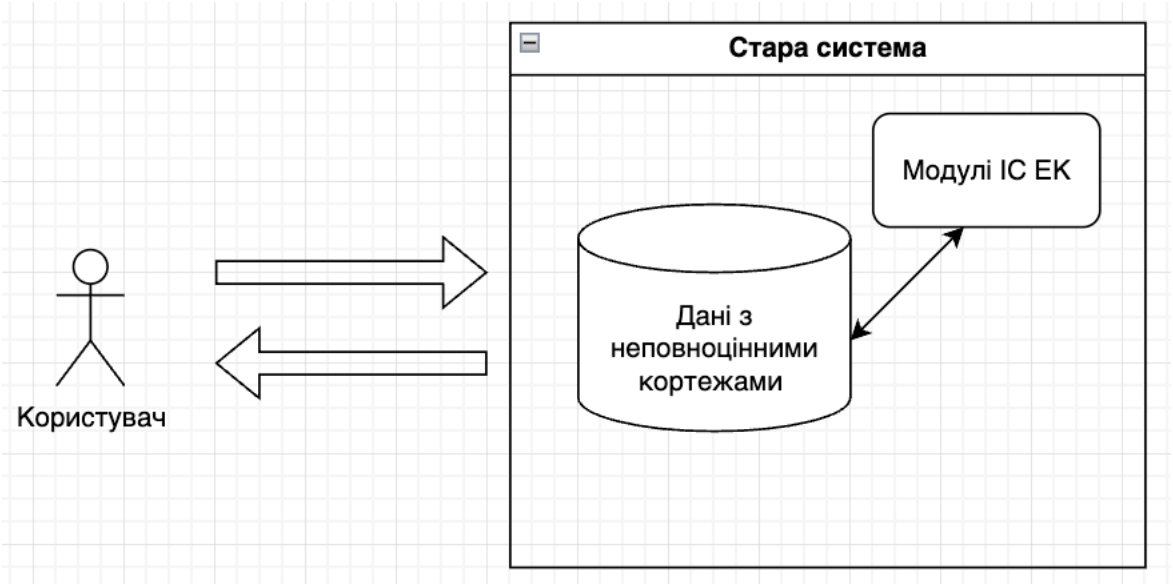


Рисунок 1.2 - Схема старої версії ІС

Нова версія системи (SAP Commerce Cloud 2211, хмарне рішення):

– архітектура: хмарна інфраструктура забезпечує централізоване адміністрування та автоматичне масштабування залежно від навантаження. Підтримка та оновлення виконуються постачальником, що значно знижує навантаження на замовника;

– база даних: таблиці мають більш детальний опис і складну структуру зв’язків, що дозволяє проводити глибокий аналіз даних, створювати точні звіти та забезпечувати більшу гнучкість у роботі з даними;

– продуктивність: вища продуктивність досягається завдяки використанню сучасного обладнання та автоматизованих механізмів оптимізації в клауді;

– можливості оновлення: оновлення системи відбуваються автоматично, що забезпечує доступ до нових функцій і захисних механізмів без переривання бізнес-процесів;

– безпека: хмарна версія пропонує розширені механізми безпеки, такі як багаторівнева автентифікація, регулярні оновлення та резервне копіювання даних у режимі реального часу.

Схема ІС наведена на рисунку 1.3.

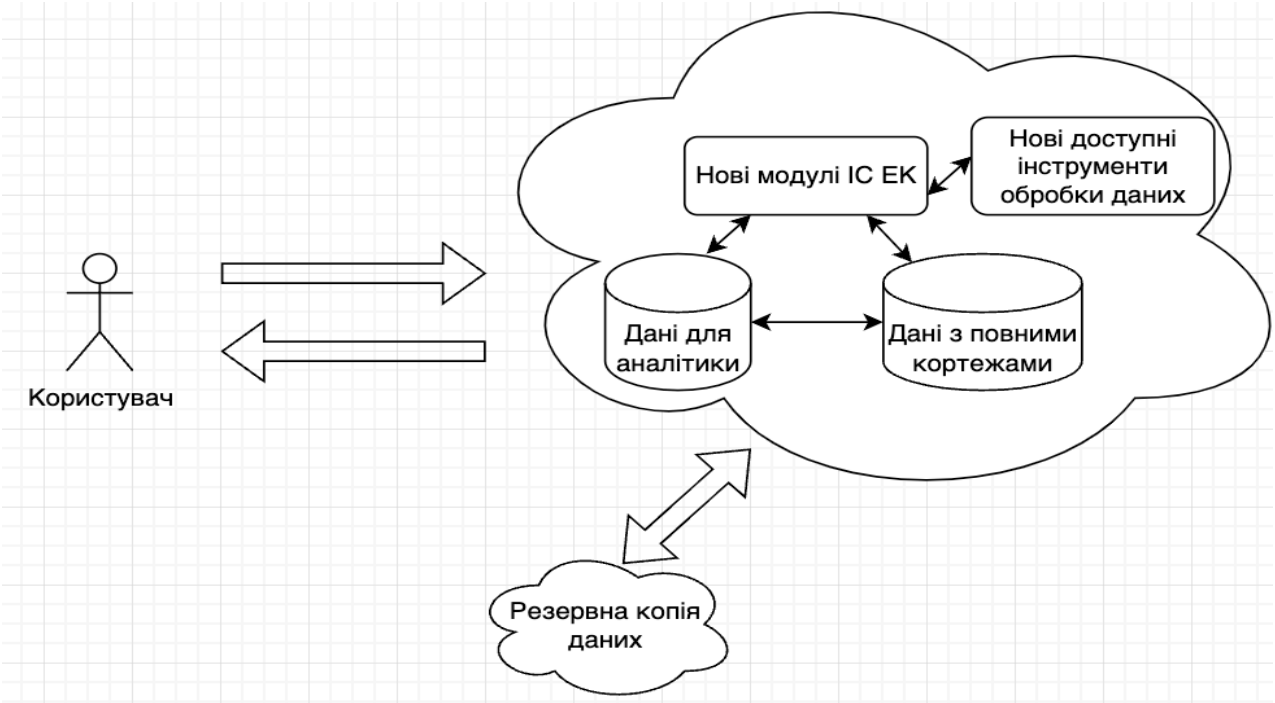


Рисунок 1.3 - Схема ІС нової версії

Ключові моменти у порівнянні двох систем наведені у таблиці 1.1.

Таблиця 1.1 - Порівняння інформаційних систем

Характеристика	Hybris 6.0 (on-premise)	SAP Commerce 2211
Інфраструктура	Локальні сервери	Хмарна інфраструктура
База даних	Спрощена структура	Деталізована структура

Кінець таблиці 1.1

Характеристика	Hybris 6.0 (on-premise)	SAP Commerce 2211
Продуктивність	Обмежена локальним обладнанням	Автоматичне масштаб. (вертикальне, горизонтальне)
Оновлення	Складний процес	Автоматичні оновлення
Безпека	Локальна відповідальність	Професійні хмарні механізми

Тобто, як можна побачити, виникає велика кількість проблем, яких можна позбутись перейшовши на нову версію продукту інформаційної системи електронної комерції:

– інфраструктура: перейшовши на нову версію можна мати підтримку хмарного середовища, що надає багато інструментів, які можна використовувати для покращення функціонування (PaaS).

– база даних: нова версія бази має більш детальний опис, більше корисних атрибутів для більш корисного аналізу дій користувачів та більш зручного способу отримання необхідних аналітичних даних.

– продуктивність: маючи в наборі інструменти, які є у хмарних рішеннях, з’являється змога покращити процес масштабування технічних потужностей.

– безпека: окрім інструментів, розміщення ІС у хмарному середовищі надає додаткові корисні механізми для забезпечення безпеки даних користувачів.

1.2 Аналіз процесів міграції даних в ІС

Розробники різноманітних програмних засобів та додатків постійно

стикаються з проблемами, що виникають через збільшення об’ємів даних та зміни у їхньому форматі. Більшу частину цих проблем вирішують за допомогою традиційних мір, наприклад - нарощенню продуктивності серверу, на якому встановлена СУБД або додаванню додаткових реплик у кластер, але у деякий момент часу може виникнути потреба у тотальній зміні сховища даних. Саме в таких випадках виникає потреба використовувати міграцію даних.

Міграція даних – процес перенесення даних з вихідної бази даних в цільову, причому їх схеми можуть відрізнятися. Причини, за якими організації починають проекти з міграції даних, може бути досить багато: від оновлень додатків і впровадження нових корпоративних систем до повномасштабної реструктуризації внаслідок злиття компаній.

Нині, не існує єдиного підходу до здійснення міграцій даних, тому дуже велика кількість програмних продуктів використовують застарілі версії СУБД, що лімітовано підтримуються, або і зовсім не підтримуються, розробниками. Також, одними з причин ступору на одній версії СУБД є те, що міграції несуть за собою дуже багато ризиків у вигляді втрати даних та несумісності існуючого програмного коду із модернізованою СУБД [1]. Все вищевказані фактори відштовхують власників програмних продуктів від міграцій подібного вигляду.

Основними факторами [2], що спонукають до міграції є:

– зменшення витрат – часто компанії переходять від локального сховища даних до хмарного, або відмовляються від дорогого ліцензування у сторону відкритих продуктів;

– модернізоване програмне забезпечення – перехід від застарілої системи до системи, розробленої для сучасних потреб. У епоху великих даних нові технології зберігання є необхідністю. Наприклад, компанія може вибрати перехід від застарілої бази даних SQL до іншої більш гнучкої системи.

– одне джерело істини – переміщення всіх даних в одне місце, доступне для всіх підсистем. Іноді це відбувається після придбання, коли системи потрібно поєднувати. Або це може статися, коли різні системи прокладаються по всій компанії. Наприклад, відділ IT може використовувати одну базу даних, тоді як група маркетингу використовує іншу базу даних, і ці системи не можуть "спілкуватися" один з одним. Якщо у вас різні бази даних, несумісні, важко отримати розуміння своїх даних.

Технічно, міграція до іншого сховища даних складається з:

– міграції схем або структур даних - структури даних у існуючому сховищі даних доведеться відтворити у новому. У більшості випадків буде доступний еквівалентний або майже еквівалентний тип даних;

– міграції даних - це процес переміщення даних з одного сховища даних в інше. Хоча, на перший погляд, це може здатися досить простим, процес може передбачати перетворення даних для їх сумісності з новим сховищем;

– міграції програм - зазвичай, сховище даних є прикладним компонентом іншою програмної системи, що власне реалізує бізнес-логіку. Тому, потрібно змінити код даної програмної системи для забезпечення сумісності з новим сховищем.

Якщо власник програмного продукту зважується на дану процедуру, то вона повинна бути непомічена для кінцевих користувачів та максимально спланована.

На перше місце ставиться два принципи - принцип якомога швидшої міграції та мінімізації внесення змін до програмного коду для роботи під новою СУБД з якомога меншою кількістю, породжених цим переходом, помилок і змін [3]. З цього загального положення безпосередньо випливає практичний висновок - наскільки можна не вносити інші, крім як

обумовлені, безпосередньо, завданнями перенесення, зміни в програмний код в процесі цього перенесення.

Невдала міграція може призвести до деградації якості або, взагалі, втрати даних. Це може статися навіть тоді, коли дані у вихідній СУБД виглядають цілком адекватними та придатними для використання. Крім того, будь-які проблеми, які існували у даних, можуть бути ампліфіковані, після їх переносу у нове сховище [4].

Зазвичай, стратегії міграції даних запобігають використанню допоміжних програмних комплексів [5], які в результаті створюють більше проблем, ніж вирішують. Плануючи та розробляючи стратегію роботи, розробникам потрібно приділяти таким міграціям всю свою увагу, не применшуючи їх значимість та об’єм.

Аналізуючи досвід міграції даних великої кількості інформаційних систем можна виділити типову процедуру міграції даних, яка містить в собі:

– аналіз форматів даних структури вихідної бази і цільової;
– підготовка плану міграції та перетворення даних;
– визначення взаємозв'язків між таблицями (ієрархії об'єктів);
– визначення послідовності перенесення даних відповідно до ієрархії залежностей [6].

Механізм міграції повинен гарантувати перегляд і перенесення всіх записів з вихідної БД, гарантувати цілісність даних, а також гарантувати, що перенесення записів буде здійснюватися з урахуванням всіх залежностей [7]. Запис може бути перенесений тільки в тому випадку, якщо він є незалежною, або записи, від яких залежить цей, вже перенесені в цільову БД. Тому основним правилом, що визначає порядок перегляду та перенесення, є залежності між записами. Залежно записів визначаються за зовнішніми ключами.

Механізм міграції даних в ІС реалізується у вигляді процедури, яка спочатку викликається для незалежних записів, а далі рекурсивно і для

залежних від цього запису [8]. Тому порядок пересування повинен починатися з таблиці, яка містить незалежні записи. Тоді основний порядок, визначений на множині таблиць схеми бази даних – залежність між таблицями. Іноді можна не враховувати взаємозв'язку, а просто відключити всі зовнішні ключі перед міграцією і перебудувати їх після завершення всіх маніпуляцій з даними. Виняток – наприклад, коли нова версія бази даних вже знаходиться в експлуатації, виконання сценарію зі зміни об'єктів в новій версії бази даних, безпосереднє перенесення даних з необхідними перетвореннями – на льоту, виконання сценарію для відновлення відключених індексів, додаткових перетворень і т.д. після завершення процедури міграції даних [9].

Зазвичай, план міграції даних (див. рис. 1.4) повинен враховувати наступні фактори [10]:

– аудит та дослідження даних та їх структури;
– обслуговування;
– журналювання.

Аудит та дослідження даних та їх структури полягає в тому, що перед міграцією вихідні дані повинні пройти повний аудит. Якщо ігнорувати цей крок, можуть з’явитися несподівані проблеми.

Обслуговування означає що якість дані може погіршуватись з плином часу, що робить їх ненадійними. Це означає, що повинні існувати засоби контролю та підтримки вищевказаної якості.

Журналювання демонструє що відстеження та звітування про якість даних є важливим, оскільки це дозволяє краще зрозуміти цілісність та якість даних.

Рисунок 1.4 - Типова схема плану міграції даних в ІС

Отже, можна сказати, що внаслідок фундаментальних архітектурних відмінностей між різними сховищами даних, міграція може бути складним завданням, що включає в себе надзвичайно багато підготовчих та виконавчих кроків пов'язаних із ризиками, таких як збереження цілісності інформації, забезпечення безпеки під час перенесення та мінімізація часу простою [11]. Наприклад, перехід із Hybris 6.0 на SAP Commerce 2211 потребує не лише технічної адаптації, але й врахування змін у бізнес-процесах. Одним із найважливіших завдань є забезпечення сумісності між старими та новими форматами даних. Використання сучасних інструментів, таких як SAP Data Services, дозволяє автоматизувати складні операції й інтегрувати різні джерела даних. Крім того, важливо враховувати питання безпеки: використання хмарних сервісів потребує шифрування даних під

час їх передачі та зберігання. Однак ними можна керувати та пом'якшувати за допомогою чітко визначеної стратегії та плану перевірки, адже ефективна міграція даних сприяє не лише технічному вдосконаленню ІС, але й відкриває нові можливості для бізнесу. Покращення якості даних, їхня інтеграція в аналітичні платформи, зменшення витрат на підтримку застарілих систем — усе це створює конкурентні переваги та підвищує адаптивність до змін ринку.

1.3 Аналіз підходів до міграції даних

Одним з основних методів перенесення даних є метод ETL – витяг, трансформація та завантаження. Це означає, що дані повинні бути вилучені з поточного сховища даних, пройти ряд функцій-трансформацій, після чого вони можуть бути завантажені в цільове місце. Типи реалізації методу ETL наведені у таблиці 1.2.

Таблиця 1.2 – Типи реалізації методу ETL

Тип реалізації	Опис
Витяг	Вилучення даних із вхідної СУБД
Трансформація	Використовується для застосування ряду правил або функцій на вхідні дані (вилучені дані), такі як об'єднання, агрегування або функції фільтру, щоб зробити дані придатними для використання.
Завантаження	Процес завантаження даних у цільову СУБД
Експорт через файли	Експорт даних до файлів CSV/JSON, їх завантаження на сервер нової бази даних та подальший імпорт до нової СУБД.

Продовження таблиці 1.2

Тип реалізації	Опис
Пряме перенесення	Перенесення даних із однієї СУБД до іншої без використання проміжних файлів шляхом запитів на вибірку, трансформацію та запис у цільову СУБД.

Схема процесу міграції шляхом експорту даних через файли формату CSV/JSON наведена на рисунку 1.5.

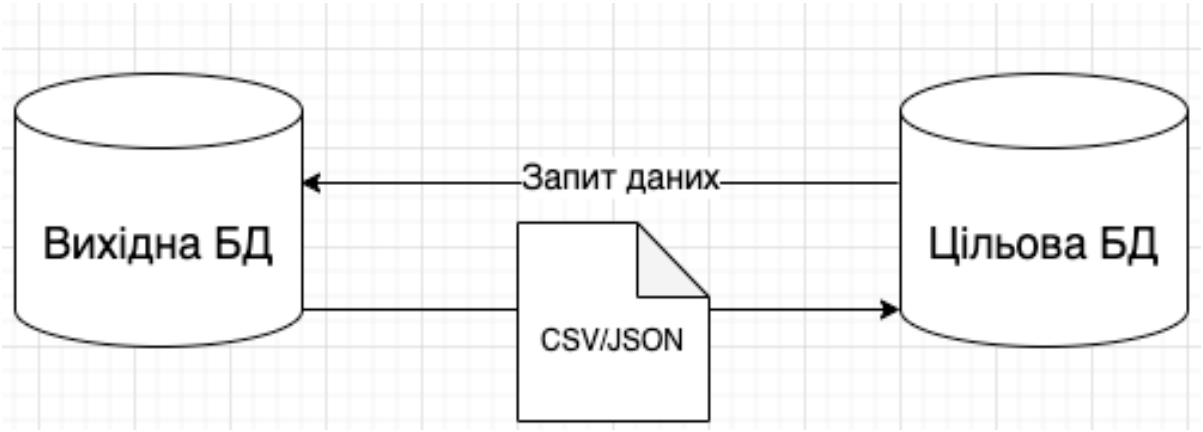


Рисунок 1.5 - Схема міграції за допомогою файлу

Іншим простим способом є створення проміжної програми. Вона повинна зв’язуватись з вихідною і цільовою базами даних і виконувати необхідні перетворення. Схема такого варіанту зображена на рисунку 1.6.

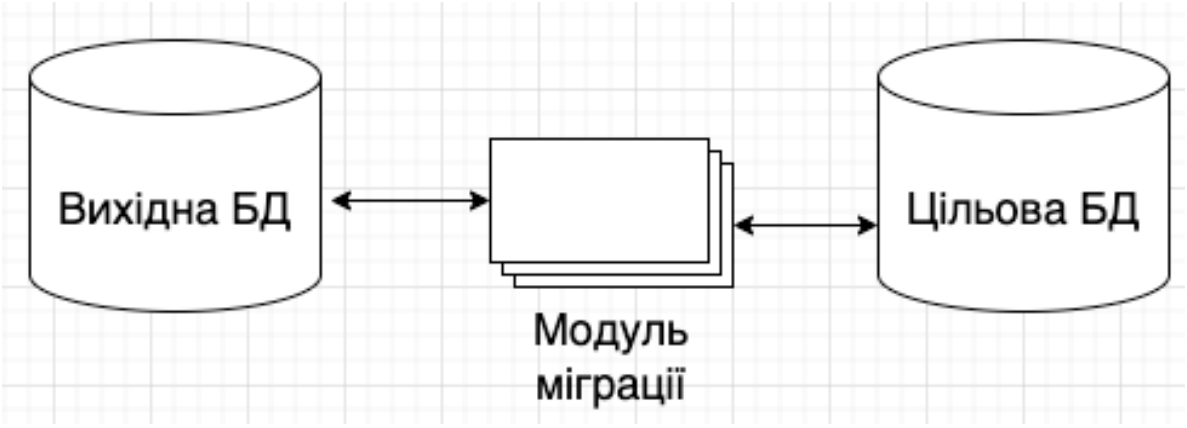


Рисунок 1.6 - Схема міграції з використанням проміжної програми

Вилучення даних. Більшість сучасних організацій не обмежується єдиним типом даних чи інформаційною системою. Як правило, вони працюють з численними джерелами даних та використовують широкий спектр інструментів для аналізу й отримання бізнес-аналітики. Для ефективного управління такими різнорідними даними важливо забезпечити їхню інтеграцію та можливість вільного переміщення між системами й додатками.

Перед тим як перемістити дані до нового сховища, необхідно вилучити їх із поточного джерела [12]. На початковому етапі процесу ETL дані, незалежно від їхньої структури, імпортуються та об'єднуються у проміжному сховищі. Виконання цієї задачі вручну може бути дуже трудомістким та спричиняти помилки, тому доцільно використовувати спеціалізовані ETL-інструменти, які автоматизують цей процес та підвищують його ефективність.

Трансформація. Цей етап передбачає застосування правил і процедур, які забезпечують відповідність даних вимогам цільового сховища. Він включає очищення даних (усунення некоректних або відсутніх значень), стандартизацію (застосування єдиних форматів), видалення дублікатів, перевірку на аномалії та сортування. Ці кроки покращують якість і цілісність даних, гарантуючи їх готовність до завантаження.

Завантаження. Завершальний етап ETL передбачає перенесення підготовлених даних у цільове сховище. Це може бути виконано двома способами: повним завантаженням (усі дані переносяться одразу, що створює ризик збільшення обсягів даних) або частковим завантаженням (дані передаються поетапно, що мінімізує ризики дублювання).

Реплікація. Реплікація даних передбачає створення ідентичних копій інформації у різних вузлах баз даних. Цей процес може здійснюватись у режимі "активний-пасивний" (дані передаються в одному напрямку) або "активний-активний" (обмін даними в обидві сторони). У випадках, коли реплікація виконується між різними СУБД, зазвичай використовуються сторонні інструменти, які забезпечують конвертацію даних.

Ручне кодування. Використання власноруч створених рішень для міграції даних може бути доцільним у невеликих проєктах із обмеженими обсягами інформації. Однак цей підхід супроводжується високими витратами часу та ресурсів, особливо у випадках масштабних міграцій. Крім того, такі рішення часто виявляються менш гнучкими порівняно з готовими ETL-інструментами.

Загалом, успішна міграція даних у "живих" системах залежить від ретельного планування, вибору оптимального підходу до синхронізації та використання спеціалізованих інструментів, таких як механізми реплікації, логування та трансляції транзакцій. Це дозволяє знизити ризики втрати даних, уникнути тривалих простоїв та забезпечити коректну роботу обох СУБД протягом всього процесу міграції.

Завдяки комбінуванню методів логування транзакцій, їхньої трансляції, реплікації та інших підходів можна адаптувати процес до конкретних потреб системи, забезпечуючи надійність і ефективність. У сучасних умовах важливість правильної організації міграції даних лише зростає, адже розвиток технологій і збільшення обсягів даних постійно ставлять перед бізнесом нові виклики. Це підкреслює необхідність не тільки

вдосконалення технічних рішень, але й розробки стратегій, які забезпечать довгострокову стабільність і адаптивність інформаційних систем.

1.4 Дослідження процесних моделей міграції даних

Згідно з опитуванням TDWI [13] , що було проведено з метою виявлення технологій, які організації використовують для міграції або консолідації баз даних, ETL (Extract-Transform-Load) став найбільш розповсюдженим підходом, і 41% респондентів проголосували за нього. Інші відповіді включали: ручне кодування (27%), реплікацію бази даних (11%) та інтеграцію корпоративних програм (3,5%) (див. рис. 1.7).

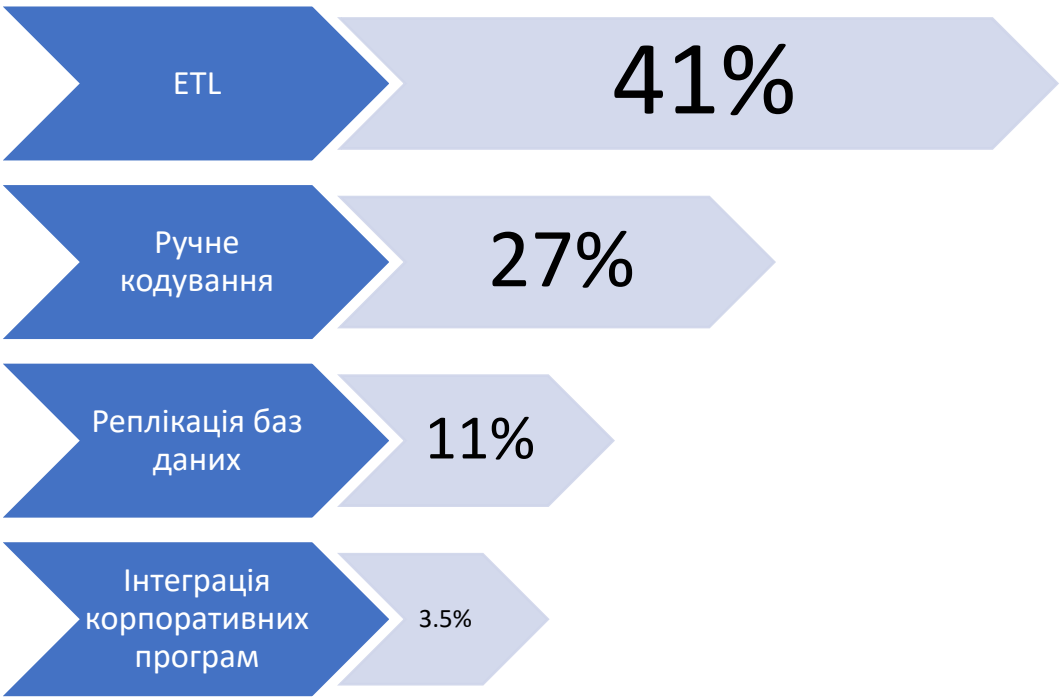


Рисунок 1.7 – Діаграма використання методів міграції

Кодування вручну є привабливим рішенням, незважаючи на, зазвичай, недостатню продуктивність та великі грошові розтрати.

Дослідження показали, що розробка та підтримка інтеграції даних на базі інструментів є набагато продуктивнішою, ніж самописні рішення. Цей метод буде існувати завжди, бо розробники не можуть відмовитись від нього, оскільки він дає дуже великий простір для свободи і є дуже гнучким [14].

Реплікація бази даних є простим і доступним варіантом, але може не відповідати деяким вимогам організації та проекту. Зазвичай інструменти реплікації є вбудованими у СУБД, але це торкається тільки сховищ розроблених одним виробником. Існують інструменти реплікації вищого класу, що забезпечують двонаправлену, неоднорідну синхронну або асинхронну синхронізацію даних, що потрібно, коли старе та нове сховище використовується одночасно [15]. Зазвичай інструменти реплікації обмежені у видозміні даних при синхронізації.

Інтеграція корпоративних програм (EAI) є одним з найкращих варіантів при швидкому переміщенні невеликих обсягів інформації між логічними рівнями додатків. Але, інструменти EAI зазвичай не можуть бути використані при міграції даних великого обсягу, з трансформацією, або профілюванням.

Інструменти ETL можуть впоратися майже із будь-якими вимогами до процесу міграції даних. Сюди входять обробка наборів великих даних, поглиблене профілювання та інтеграція між кількома платформами. Деякі засоби ETL навіть надають можливість автоматизації стандартних завдань ETL, таких як отримання даних з специфічних сервісів, їх перетворення в уніфікований формат та завантаження в цільову базу даних. Виходячи із аналізу, для проведення експериментального дослідження були обрані 3 найпопулярніших метода міграції – ETL, ручне кодування та реплікація.

У таблиці 1.2 наведено приклади моделей процесів міграції даних.

Таблиця 1.2 – Приклади моделей процесів міграції даних

Модель	Переваги	Недоліки
ETL	Універсальність, підходить для більшості проєктів; забезпечує високу якість і консистентність даних	Висока тривалість процесу; можливі затримки при великих обсягах даних
Реплікаційна модель	Забезпечує безперервну роботу системи; підходить для масштабних міграцій.	Висока потреба в ресурсах; складність налаштування активний-активний режимів
Транзакційна синхронізація	Мінімізація простоїв системи; дані синхронізуються у реальному часі	Висока складність реалізації; можливі затримки при обробці великих обсягів даних
Перехресна реплікація	Можливість міграції між різними типами СУБД; забезпечує сумісність через сторонні інструменти	Залежність від сторонніх інструментів; ускладнене налаштування та підвищені вимоги до інтеграції
Ручне кодування	Можливість створення унікального рішення для специфічних потреб; мінімальні залежності від сторонніх інструментів	Дуже трудомісткий процес; висока ймовірність помилок; складність масштабування для великих обсягів даних

1.5 Постановка задачі дослідження

Актуальність дослідження полягає у наступному. Існуючі процесні моделі міграції даних в інформаційних системах електронної комерції стикаються з низкою викликів, таких як забезпечення безперервності роботи систем, підтримка узгодженості даних, адаптація до нових архітектур, а також ефективне використання інструментів ETL та реплікації. Усе це ускладнюється збільшенням обсягів даних і необхідністю враховувати індивідуальні особливості інформаційних систем. Традиційні методи, які використовуються для міграції, потребують ручного визначення відповідності даних людми-експертами в предметній галузі, що призводить до значних витрат часу та збільшення бюджету проектів.

Об’єктом дослідження є процес міграції даних в інформаційних системах електронної комерції.

Предметом дослідження є методи та процесні моделі міграції даних в інформаційних системах електронної комерції.

Метою даної роботи є дослідження та удосконалення процесів міграції даних для підвищення ефективності інформаційних систем електронної комерції на основі використання сучасних інструментів, запитів до великих мовних моделей та використання процесних моделей.

Для досягнення мети дослідження необхідно вирішити такі задачі:

– розробка запитів щодо міграції даних до великих мовних моделей;

– удосконалення процесної моделі міграції даних;

– розробка інформаційної технології міграції даних в системах електронної комерції на основі вдосконаленого методу запитів щодо міграції даних до великих мовних моделей;

– експериментальна перевірка процесної моделі міграції даних в інформаційних системах електронної комерції.

2 РОЗРОБКА ПРОЦЕСНОЇ МОДЕЛІ МІГРАЦІЇ ДАНИХ В ІС ЕЛЕКТРОННОЇ КОМЕРЦІЇ З ВИКОРИСТАННЯМ ЗАПИТІВ ДО ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

2.1 Розробка запитів щодо міграції даних до великих мовних моделей

Під час міграції даних між базами даних вихідної та цільової систем виникає завдання співставлення таблиць і атрибутів, структура яких може відрізнятися. [16] Це є важливим етапом для забезпечення коректного перенесення інформації та її узгодженості в цільовій системі.

Основні етапи зіставлення:

– аналіз структури вихідної бази даних;
– аналіз структури цільової бази даних;
– співставлення таблиць і атрибутів;
– формування зіставлення.

Аналіз структури вихідної бази даних складається з етапу збору інформації про таблиці, атрибути, їх типи даних, зв’язки між таблицями та інші характеристики. Цей процес включає отримання метаданих:

$$S = \{(T, A, D) \mid T \in Tables, A \in Attributes(T), D \in DataTypes(A)\},$$

де S — множина метаданих;

T — таблиці;

A — атрибути;

D — типи даних атрибутів.

Аналіз структури цільової бази даних - аналогічно вихідній системі, отримуються метадані цільової бази даних.

Співставлення таблиць і атрибутів полягає в тому, що для кожної таблиці у вихідній базі даних визначається відповідна таблиця у цільовій базі. Для кожного атрибута таблиці вихідної бази обирається атрибут у цільовій таблиці на основі:

– назви,

– типу даних,

– контексту використання.

Формування зіставлення полягає в тому, що генерується остаточний результат відповідностей, як у таблиці 2.1.

Таблиця 2.1 – Таблиця відповідностей

Таблиця (віхідна)	Таблиця (цільова)	Атрибут (вихідний)	Атрибут (цільовий)	Тип даних
Orders	SalesOrders	OrderId	ID	Integer
Orders	SalesOrders	CustomerID	ClientID	String
Orders	SalesOrders	OrderDate	PurchaseDate	DateTime

У зв’язку з прямим зіставленням назв таблиць, кортежей, їхніх типів, виникають виклики, які зазвичай вирішують відомими та популярними способами [17]:

– різниця у назвах: використання словників відповідностей для стандартних назв таблиць і атрибутів;

– різниця у типах даних: підготовка спеціальних правил трансформації даних для приведення типів до єдиного формату;

– складні зв’язки між таблицями: використання алгоритмів аналізу зв’язків (наприклад, зовнішніх ключів) для коректного перенесення пов’язаних даних.

Розробка програмного модуля для зіставлення таблиць і атрибутів баз даних дозволяє підготувати основу для коректного перенесення даних. Це

включає аналіз метаданих обох баз, створення відповідностей і підготовку правил трансформації, що є критично важливим етапом у процесі міграції даних. На рисунку 2.1 наведена схема такого підходу, яка складається з 6 етапів: аналіз структури вихідної БД, аналіз структури цільової БД, співставлення таблиць та атрибутів, аналіз типів даних атрибутів, співставлення атрибутів за назвами (з урахуванням типів), формування таблиці відповідностей.

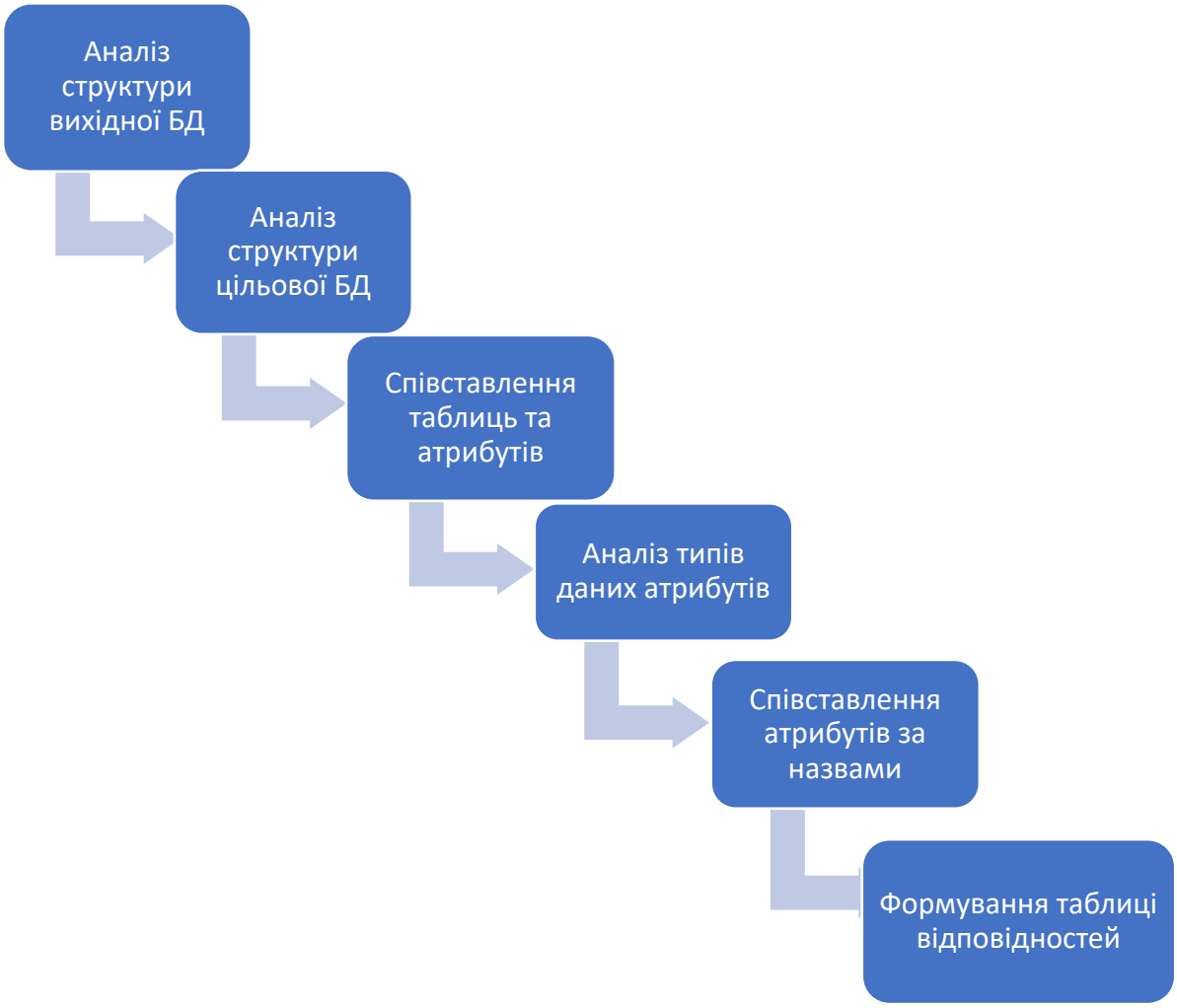


Рисунок 2.1 – Схема процесної моделі міграції з прямим зіставленням

Рисунок 2.1 надає розуміння загального підходу до міграції та визначає основні етапи цього підходу:

– аналіз структури вихідної бази даних: цей етап включає вилучення інформації про структуру вихідної бази даних. Збираються назви таблиць, атрибутів, їх типи даних, а також зв’язки між таблицями (наприклад, зовнішні ключі). На цьому етапі також можуть бути враховані обмеження бази даних, такі як унікальність чи обов’язковість значень. Приклад: Таблиця Orders має атрибути OrderID (Integer), CustomerID (String), OrderDate (DateTime);

– аналіз структури цільової бази даних: проводиться аналогічний збір інформації для цільової бази даних. Вивчається, як структуровані таблиці та атрибути, та чи існують схожі або відповідні елементи. Враховуються специфічні особливості цільової бази, наприклад, ієрархія таблиць, типи даних, синтаксис чи правила іменування. Приклад: Таблиця SalesOrders містить атрибути ID (Integer), ClientID (String), PurchaseDate (DateTime);

– співставлення таблиць та атрибутів: виходячи з отриманих структур обох баз даних, для кожної таблиці вихідної бази шукається відповідна таблиця у цільовій базі. Співставлення базується на схожості назв таблиць, їх контексту та логіки використання в системі. Можуть застосовуватись попередньо визначені словники відповідностей або алгоритми порівняння текстів. Приклад: Таблиця Orders співставляється з таблицею SalesOrders. Для кожного атрибута у вихідній таблиці визначаються відповідні атрибути у цільовій таблиці. Порівнюються назви, типи даних, розмірність (наприклад, довжина строкових значень) та логічна структура. У разі невідповідності типів даних можуть створюватись правила трансформації, наприклад, конвертація числового значення в текстове або перетворення дати. Приклад: OrderID співставляється з ID. CustomerID співставляється з ClientID. OrderDate співставляється з PurchaseDate;

– аналіз типів даних атрибутів: проходить аналіз кожного атрибуту для знаходження потрібного за типом на призначенням атрибуту;

– співставлення атрибутів: після аналізу здійснюється створення словнику / таблиці для створення технічної документації, за планом якої і відбуватиметься міграція;

– формування таблиці відповідностей: на основі попередніх етапів створюється таблиця, яка відображає всі знайдені відповідності між таблицями та атрибутами. Таблиця включає також правила трансформації даних для тих випадків, коли типи даних або формат атрибутів відрізняються. Цей етап завершує процес підготовки до міграції, оскільки забезпечує повне розуміння зв’язків між вихідною та цільовою структурами.

Проте враховуючи той факт, що дана процесна модель базується на самостійному ручному співставленню атрибутів, можна дійти висновку що це викликає потенційні проблеми, так як існує людський фактор. Більш того, це вимагає велику кількість часу для аналізу всіх таблиць, які будуть мігрувати. Зазвичай, ІС електронної комерції мають досить велику кількість таблиць, і будь-яка помилка при співставленні або аналізі має великі наслідки як для бізнесу, так і для користувачів.

З урахуванням впровадження великих мовних моделей для обробки даних та аналізу великої кількості даних та витратою великої кількості часу на аналіз об’єктів що мігрують, новий етап з запитом до мовних моделей для співставлення атрибутів, таблиць з урахуванням всіх тонкощів та потенційних проблем може автоматизувати та спростити як аналіз всіх баз даних, так і допоможе уникнути помилок під час зіставлення таблиць та атрибутів за назвами, типами тощо.

2.2 Удосконалена процесна модель міграції даних

Відповідний етап із запитом до великої мовної моделі щодо міграції даних допоможе автоматизувати етапи співставлення таблиць та атрубутів, аналізу типів даних атрибутів, співставлення атрибутів за назвами та типами.

На рисунку 2.2 наведено відповідну оновлену схему з додаванням нового етапу.

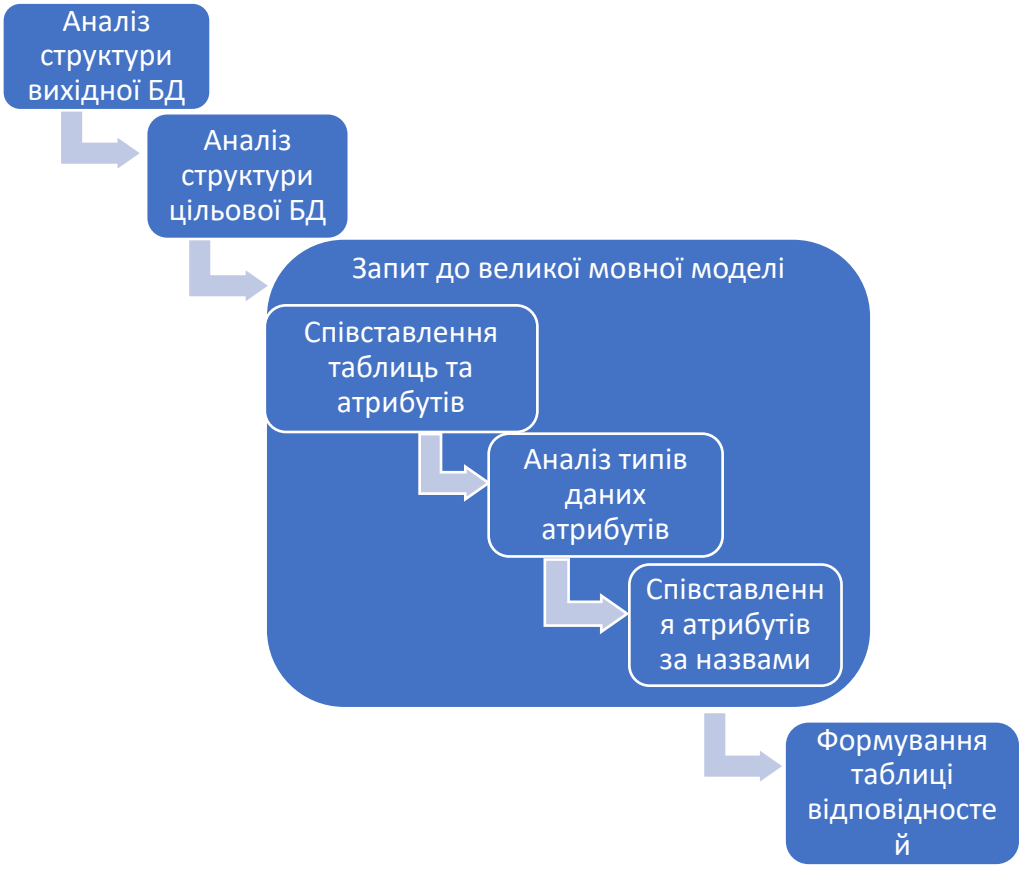


Рисунок 2.2 – Оновлена схема процесної моделі міграції з прямим зіставленням

В рамках розробки програмного модуля для міграції даних передбачається інтеграція LLM для:

– автоматичного аналізу назв таблиць та атрибутів у вихідній базі даних;

– визначення найбільш ймовірних відповідностей між вихідними та цільовими елементами;

– формування пропозицій щодо співставлення, які можуть бути затверджені або скориговані користувачем.

Алгоритм роботи:

– підготовка даних для аналізу;

– запит до великих мовних моделей;

– результат співставлення;

– автоматична інтеграція.

Підготовка даних для аналізу полягає в тому, що з бази даних вилучаються назви таблиць, атрибутів і описів. Ці дані передаються у вигляді запиту до великої мовної моделі. Формула для вилучення:

$$\boldsymbol{D = \{(T, A) \mid T \in Tables, A \in Attributes(T)\}},$$

де D – множина даних для аналізу;

T – назви таблиць;

A – атрибути таблиць.

Запит до великих мовних моделей виконується шляхом текстового запиту до LLM, який формується на основі вилучених даних.

Аналіз та співставлення полягає в тому, що LLM повертає набір відповідностей:

$$\boldsymbol{M = \{(Ts, Tt, As, At)\}},$$

де Ts і As – таблиці та атрибути вихідної бази;

Tt і At – відповідні таблиці та атрибути цільової бази.

Результат співставлення – це результати аналізу LLM можуть бути подані у вигляді списку або візуалізовані в інтерфейсі для перевірки й затвердження користувачем.

Автоматична інтеграція полягає в тому, що на основі затверджених співставлень створюються SQL-скрипти для трансформації та перенесення даних між базами.

Приклади текстового запиту до LLM наведені на рисунку 2.3.

– *«Ти фахівець з міграції баз даних, проаналізуй структуру таблиці <НАЗВА ТАБЛИЦІ> у вихідній базі даних та знайди відповідну таблицю у вихідній базі даних. Опціонально: використовуй контекст минулих запитів, наведи приклади таблиць зі схожою структурою»*

– *«Є таблиця <НАЗВА ТАБЛИЦІ ВИХІДНА> із атрибутами <ПЕРЕЛІК АТРИБУТІВ> у вихідній базі даних. У цільовій базі даних є таблиця <НАЗВА ТАБЛИЦІ ЦІЛЬОВА>. Визначте, які атрибути таблиці <НАЗВА ТАБЛИЦІ ВИХІДНА> відповідають атрибутам таблиці <НАЗВА ТАБЛИЦІ ЦІЛЬОВА>».*

Рисунок 2.3 – Патерни запиту до LLM

Модель аналізує структуру та повертає відповідність, приклад якої наведений у таблиці 2.2.

Таблиця 2.2 – Приклад результату відповідностей

Вихідна таблиця	Цільова таблиця	Вихідний атрибут	Цільовий атрибут
Orders	SalesOrders	OrderID	ID
Orders	SalesOrders	CustomerID	ClientID
Orders	SalesOrders	OrderDate	PurchaseDate

Інтеграція великих мовних моделей у процес міграції даних дозволяє автоматизувати співставлення таблиць і атрибутів, скорочуючи час на налаштування та підвищуючи точність. Такий підхід забезпечує гнучкість, зменшує ризик помилок і відкриває нові можливості для роботи з великими базами даних.

3 ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ МІГРАЦІЇ ДАНИХ В СИСТЕМАХ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Запропонований етап зі створенням запиту до великих мовних моделей щодо міграції даних у процесній моделі прямого зіставлення загалом складається з 6 компонентів:

– створення промтів для запиту;
– виконання запиту;
– обробка результату;
– формування відповідностей;
– автоматизація складних відповідностей;
– підготовка до затвердження.

На першому етапі відбувається формулювання текстових запитів, які передаються великим мовним моделям. Задля створення специфічного, найбільш точного запиту використовується технологія промтів (запитів), які містять додаткову інформацію, яка допоможе отримати найбільш точний і правильний, очікуваний результат. Промти включають специфічну інформацію про таблиці та атрибути вихідної і цільової баз даних. Мета даного етапу - створити чіткий і контекстуальний запит для отримання максимально релевантного результату.

Другий етап – виконання запиту. На цьому етапі запит передається до великої мовної моделі використовуючи API. У разі успіху та правильно побудованого запиту модель аналізуватиме структури баз даних, контекст запиту та повертатиме запропоновані відповідності.

На наступному етапі відбувається ручна обробка складних відповідностей запиту. Отриманий результат перевіряється на узгодженість із правилами відповідності, які були визначені заздалегідь.

На четвертому етапі формується таблиця відповідностей між вихідною та цільовою базами даних на основі результату запиту. У таблиці можуть зазначатись додаткові рекомендації щодо трансформації даних.

П’ятий етап виконується лише у разі присутності складних відповідностей, наприклад, об’єднання кількох атрибутів у вихідній базі в один атрибут у цільовій базі. В такому випадку модель враховує всі дані які були оброблені вручну на третьому етапі.

Фінальний етап – підготовка до затвердження. Результати представляються користувачу для перевірки й затвердження. І після цього дані використовуються для створення скриптів / логіки міграції.

Як можна побачити на рисунку 2.2, запит до великих мовних моделей об’єднує в собі декілька етапів, так як саме ці етапи він спрощує та автоматизує, тим самим створюючи таблиці відповідностей набагато швидше та якісніше. Етапи які спрощує підхід із запитами до великих мовних моделей:

– співставлення таблиць: моделі автоматично знаходять відповідності між таблицями вихідної і цільової бази, що скорочує час аналізу;

– співставлення атрибутів: автоматичний підбір атрибутів із врахуванням їх назв, типів даних і контексту;

– аналіз типів даних атрибутів: із-за великої кількості таблиць та атрибутів, а також специфіки електронної комерції людині набагато трудніше визначати та аналізувати весь список атрибутів, тому потужні сучасні моделі допомагають в цьому і надають користувачу варіанти у випадку декількох варіацій;

– співставлення атрибутів за назвами та типами: модель може запропонувати декілька варіантів співставлень, які можуть бути правильними, і надані для вирішення користувачу;

– усунення помилок: таким чином знижується ризик людських помилок при ручному зіставленні великих наборів даних;

– підготовка скриптів міграції: на основі результатів співставлення можна автоматично створити скрипти для перенесення даних, або сформувати вимоги для створення логіки міграції цих даних.

Для формалізованого подання запропонованого етапу створення запиту до великих мовних моделей щодо міграції даних було побудовано відповідну декомпозовану модель у нотації IDEF0, що зображена на рисунку 3.1.

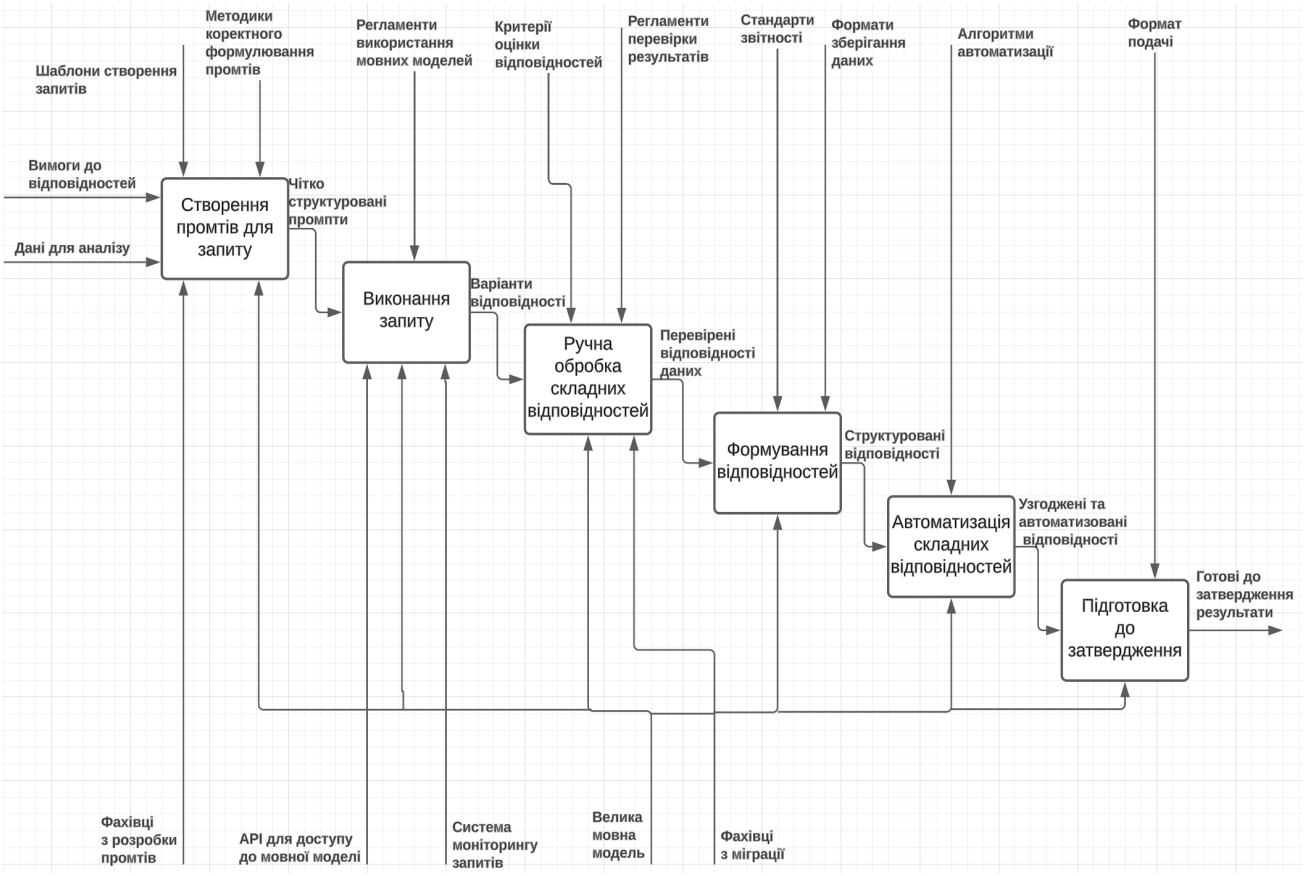


Рисунок 3.1 – Декомпозована процесна модель міграції з етапом створення запиту до великих мовних моделей у нотації IDEF0

У результаті впровадження ми отримуємо відповідну інформаційну систему, що призначена для міграції даних з можливістю автоматичного

співставлення таблиць вихідної та цільової баз даних за допомогою запиту до великих мовних моделей.

4 ПРАКТИЧНА РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ПЕРЕВІРКА ПРОЦЕСНОЇ МОДЕЛІ МІГРАЦІЇ ДАНИХ В ІС ЕЛЕКТРОННОЇ КОМЕРЦІЇ

4.1 Розробка підсистеми міграції даних системи електронної комерції

Основною метою практичної реалізації є створення програмного модуля, який дозволяє автоматизувати процес співставлення таблиць і атрибутів баз даних шляхом використання запитів до великих мовних моделей (LLM). Це сприятиме зменшенню кількості ручної роботи, підвищенню точності співставлень і скороченню часу на підготовку до міграції даних.

Розглянемо приклад реалізації процесу міграції на існуючій ІС електронної комерції, яка працює на системі SAP Commerce. В даному випадку міграція даних забезпечується за допомогою програмного рішення, реалізованого командою розробників.

Загалом, логіка міграція створена задля переносу вже існуючих реальних даних користувачів на більш сучасні Cloud рішення (SAP Cloud). В основному використовується програмна мова Java версії 17 [18], так як вона є надійною та зручною мовою для реалізації довгострокових, безпечних та масштабованих ІС.

Маючи доступи до Cloud машин, які вже мають розгорнуті версії розробленої логіки міграції, можна запустити процес міграції. Він складається з декількох фаз, які йдуть у суворо визначеному порядку:

– prefetch;
– main migration process.

Prefetch фаза є першою, адже вона відповідає за створення та відправку HTTP запитів на сервери, які взаємодіють з клієнтами. Запити відправляються з метою отримання кількості сутностей, необхідних для міграції, і також створення спеціальних записів, які будуть

використовуватись далі. На рисунку 4.1 зображено варіант налаштування цієї фази. Також на рисунку 4.2 можна побачити записи, які створюються у разі успішного результату. На цьому прикладі зображений запис який використовуватиметься для міграції моделі клієнта (customer).

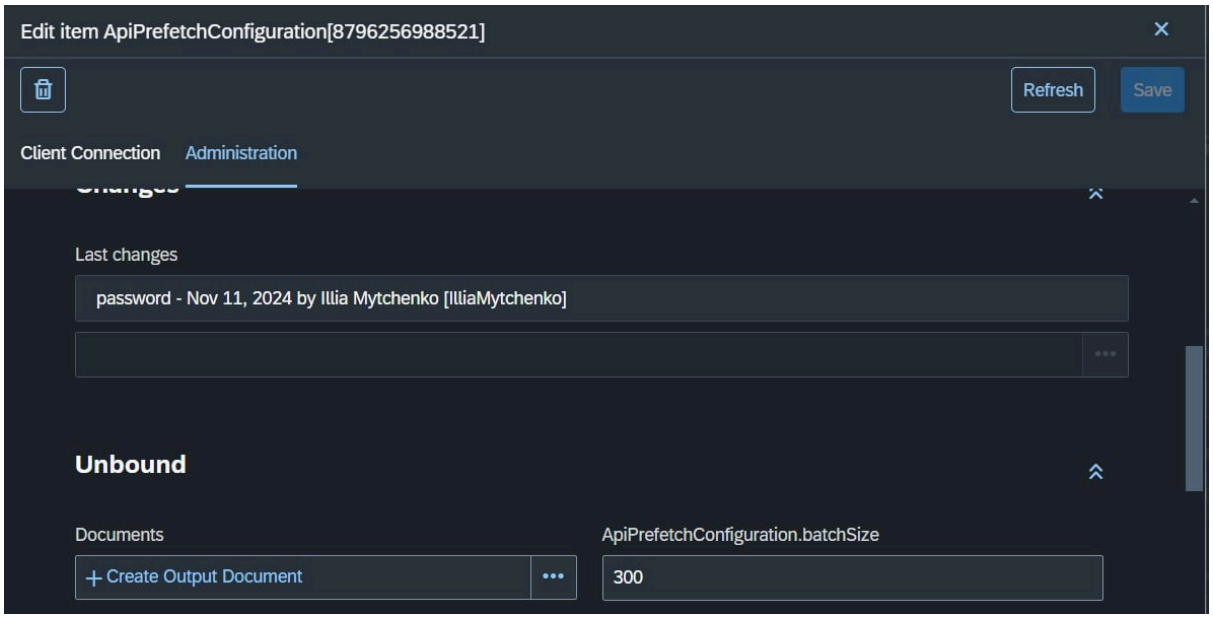


Рисунок 4.1 - Налаштування Prefetch фази

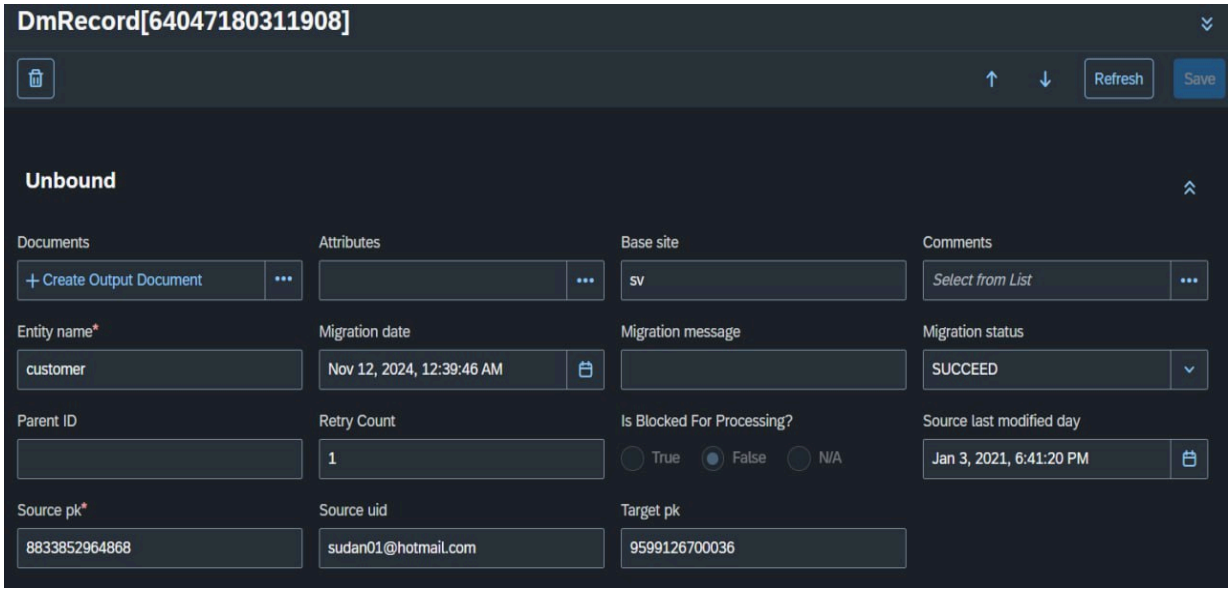


Рисунок 4.2 - Приклад запису для міграції клієнта (customer)

Main migration process (головний міграційний процес) - ця фаза є головною, так як вона використовує вже новостворені записи моделей, які треба змігрувати, та надсилає вже детальні HTTP запити з метою отримання детальної інформації і даних для міграції. На рисунку 4.3 зображено налаштування цієї фази.

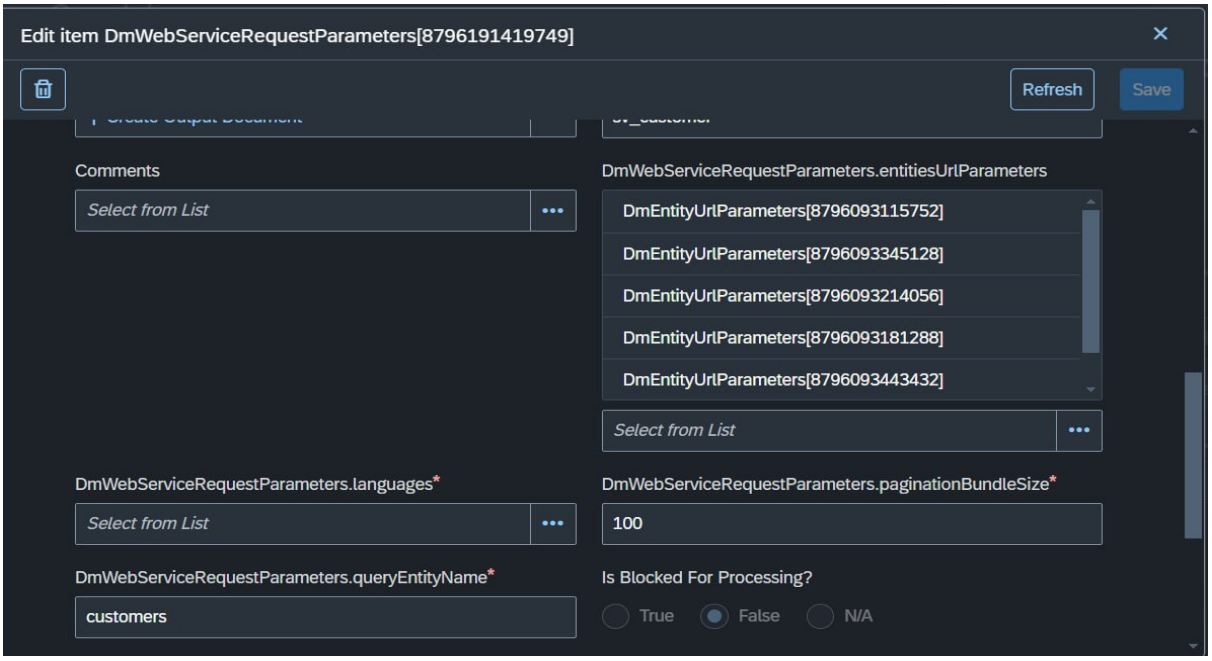


Рисунок 4.3 - Конфігурація головної фази міграції

Як можна побачити на рисунку, в налаштуваннях вказується тип моделей, які треба змігрувани і список параметрів (DmEntityUrlParameters), які використовуватимуться в HTTP запитах. Ці параметри треба налаштовувати дуже чітко і прискіпливо перевіряти, адже за цими параметрами вирішується які саме дані будуть отримані. На рисунку 4.4 продемонстрований код, який відповідає за відправку необхідних HTTP запитів.

```
protected Map getEntityData(URI entityURI, WebServiceConfigurationModel wsConfig) {
    DataMigrationRequestBuilder builder = new DataMigrationRequestBuilder()
            .entityUri(entityURI)
            .username(wsConfig.getUsername())
            .password(wsConfig.getPassword())
            .language(DataMigrationLanguageThreadLocal.getIsocode())
            .fallbackLanguages(wsConfig.getFallbackLanguages());
    return dataMigrationClient.getForEntity(builder.build());
```

Рисунок 4.4 - Логіка, яка відповідає за відправку HTTP запитів

Якщо запити повернулись успішними (HTTP Status Code = 200), то це означає що необхідні дані були успішно зчитані та отримані.

Далі, маючи необхідні дані, можна використовувати логіку задля створення необхідного об’єкту, що мігрується, і встановлювати ці самі значення у правильні кортежі. Звісно, логіка створення та встановлення значень пов’язана з роботою з БД, але в даному прикладі (SAP Commerce) існує прошарок ORM, який дозволяє працювати з класами моделі використовуючи мову Java, і в кінці автоматично перетворюючи ці класи на спеціальні конкретні SQL запити. Приклад коду який відповідає за створення об’єкту класу моделі наведений на рисунку 4.5.

```
private T createEntityModel(Class<T> entityClass) {
    return modelService.create(entityClass);
}
```

Рисунок 4.5 - Код створення конкретного об’єкту класу моделі

Враховуючи попередні приклади, можна зробити висновок, що мігрується модель класу Customer. Після створення об’єкту класу моделі треба встановити значення, отримані у HTTP Response. Ця логіка реалізована у класах-стратегіях. У цьому конкретному прикладі буде

викликатись певна стратегія, яка базується на класі моделі і буде використана для детального встановлення значень отриманих раніше.

В даному випадку використовуватиметься стратегія для міграції клієнтів (customers). Приклад коду, який реалізує логіку міграції та встановлення певних даних у відповідні атрибути класу зазначена на рисунку 4.6.

```
@Override
protected void postHandleParsedData(Map entityData, CustomerModel customer, DmRecordModel dmRecord,
                                    WebServiceConfigurationModel wsConfig) {
    setSiteContext(customer);
    setDefaultUserGroup(customer);
    setCurrency(customer, entityData);
    setLanguage(customer, entityData);
    updateSourceLastModifiedDate(dmRecord, (String) entityData.get(MODIFIEDTIME));
    attachAndSaveAddresses(entityData, customer, wsConfig);
    setLoyaltyInfo((Map) entityData.get(getLoyaltyInfoField()), customer, wsConfig);
    attachAndSaveWishlists((Map) entityData.get(WISHLIST_FIELD), customer, wsConfig);
    attachAndSavePersonalInformation(entityData, customer, wsConfig);
    if (getSessionService().isImportOrdersWithCustomer()) {
        attachAndSaveOrders((Map) entityData.get(ORDERS_FIELD), customer, wsConfig);
    }
    attachAndSaveCrmInformation(entityData, customer);
    attachAndSaveGdprInformation(entityData, customer);
    setTitle(entityData, customer);
    attachUserGroups(entityData, customer);
}
```

Рисунок 4.6 - Реалізація класу-стратегії для моделі клієнтів (customers)

На рисунку 4.7 зображений приклад коду який вручну співставляє назви атрибутів для міграції та мігрує дані відповідно до аналізу.

```
@Override
void performPopulation(Map<String, Object> entityData, CustomerModel customer) {
    if (customer.getItemModelContext().isNew()) {
        setEntityAttributeValue(customer, "creationtime", toDate(entityData.get("creationtime")));
    }
    Optional<CustomerType> customerType = getCustomerType(entityData);
    setEntityAttributeValue(customer, "type", customerType);
    setCustomerName(entityData, customer, customerType);
    setEntityAttributeValue(customer, "passwordEncoding",
            EntityParser.toString(entityData.get("passwordEncoding")));
    setEntityAttributeValue(customer, "encodedPassword", EntityParser.toString(entityData.get("encodedPassword")));
    setEntityAttributeValue(customer, LAST_PASSWORD_UPDATE,
            EntityParser.toDate(entityData.get("igcLastPwdUpdate")));
    setEntityAttributeValue(customer, "loginDisabled", toBoolean(entityData.get("loginDisabled")));
    setEntityAttributeValue(customer, "lastLogin", toDate(entityData.get("lastLogin")));
    setEntityAttributeValue(customer, "ageVerified", toBoolean(entityData.get("verifiedAgeRestricted")));
    setEntityAttributeValue(customer, HMC_LOGIN_DISABLED, EntityParser.toBoolean(entityData.get(HMC_LOGIN_DISABLED)));
    customer.setCustomerID(getCustomerId(entityData, customer));
    customer.setUid(getFieldWithoutSpaces(entityData, UID_ATTRIBUTE));
    customer.setOriginalUid(getFieldWithoutSpaces(entityData, ORIGINAL_UID_ATTRIBUTE));
}
```

Рисунок 4.7 – Реалізація ручного співставлення атрибутів таблиці

Як можна наочно побачити, назви атрибутів строго внесені до коду, що вносить жорсткий коректив до міграції. У випадку зміни якоїсь із назв атрибутів у вихідній або цільовій таблиці, ці назви потрібно також змінювати і в коді, що приведе до додаткового тестування і розгортки, які вимагають великих ресурсів.

Саме тому, можна позбавитись цих недоліків ручного прямого співставлення назв таблиць і атрибутів, додавши етап із запитом щодо міграції до великої мовної моделі.

Тому виникає потреба у створенні автоматичного запиту до великої мовної моделі задля автоматизованого співставлення таблиць та їхніх атрибутів. Приклад створення такого запиту наведений на рисунках 4.8 та 4.9.

```
protected Map<String, String> getTableAttributeMapping(URI llmApiUri, Map<String, Object> inputData, WebServiceConfigurationModel wsConfig) {

    DataMigrationRequestBuilder builder = new DataMigrationRequestBuilder()
            .entityUri(llmApiUri)
            .username(wsConfig.getUsername())
            .password(wsConfig.getPassword())
            .language(DataMigrationLanguageThreadLocal.getIsocode())
            .fallbackLanguages(wsConfig.getFallbackLanguages());

    builder.setPayload(inputData);

    Map<String, String> response = dataMigrationClient.getForEntity(builder.build());

    return response;
}
```

Рисунок 4.8 – Реалізація запиту до БД для отримання всіх даних таблиць.

```
@Override
protected void postHandleParsedData(Map entityData, CustomerModel customer, DmRecordModel dmRecord,
                                    WebServiceConfigurationModel wsConfig) {

    URI llmApiUri = URI.create("https://api.openai.com/v1/engines/davinci-codex/completions");
    Map<String, Object> inputData = Map.of(
            k1: "sourceTables", entityData.get("sourceTables"),
            k2: "targetTables", entityData.get("targetTables")
    );
    Map<String, String> mapping = getTableAttributeMapping(llmApiUri, inputData, wsConfig);

    if (mapping.containsKey("Orders")) {
        attachAndSaveOrders((Map) entityData.get(mapping.get("Orders")), customer, wsConfig);
    }
}
```

Рисунок 4.9 – Реалізація міграції з використанням запиту до великої мовної моделі

На рисунку 4.8 продемонстрований інший запит, який використовує HTTP протокол для передачі даних між ІС та БД, з метою отримати опис всіх сутностей баз даних, для подальшого співставлення та використання цих даних у запиту до великої мовної моделі.

На рисунку 4.9 наведений приклад коду зі створенням запиту до великої мовної моделі («https://api.llm-example.com/match – приклад посилання до API мовної моделі) з використанням вже отриманих даних з усіх таблиць, які беруть участь у міграційних процесах [19]. Як можна

побачити, код із прямим зіставленням полів вже не потрібен, так як велика мовна модель вже згенерувала чіткі співставлення таблиць, атрибутів за їхніми назвами та типами. На рисунку можна побачити як створюється запит до мовної моделі, передаючи URL API, вхідні дані для запиту. Результат отримуємо у вигляді мапи відповідностей між таблицями та атрибутами.

Інтеграція з великою мовною моделлю дозволяє отримати точні результати співставлення, полегшує роботу розробникам, оскільки більша частина співставлення виконується автоматично, а також зменшує кількість помилок, які можуть виникнути під час ручного аналізу. Тому, цей підхід не тільки допомагає позбутись зайвої логіки, а також сприяє поліпшенню якості роботи ІС та пришвидшенню міграційних процесів що позитивно впливає на досвід бізнесу та користувачів.

4.2 Експериментальна перевірка процесної моделі міграції даних

Основною метою експериментальної перевірки є оцінка ефективності використання великих мовних моделей (LLM) для автоматизації співставлення таблиць і атрибутів у базах даних. Для цього проведено порівняння між результатами ручного співставлення та результатами, отриманими за допомогою автоматизації із застосуванням промптів до LLM.

Ручне співставлення будемо робити вручну, визначати відповідності між таблицями та атрибутами шляхом порівняння їх назв, типів даних і логіки використання.

Запит до великої мовної моделі щодо міграції даних будемо робити сформованим текстовим промптом, який передаватиметься мовній моделі

(GPT-4), а результати аналізу і співставлення будуть отримані у вигляді таблиці відповідностей.

Звичайно, для проводження експерименту потрібно мати набір тестових даних. На рисунку 4.10 наведена фізична схема вихідної бази даних.

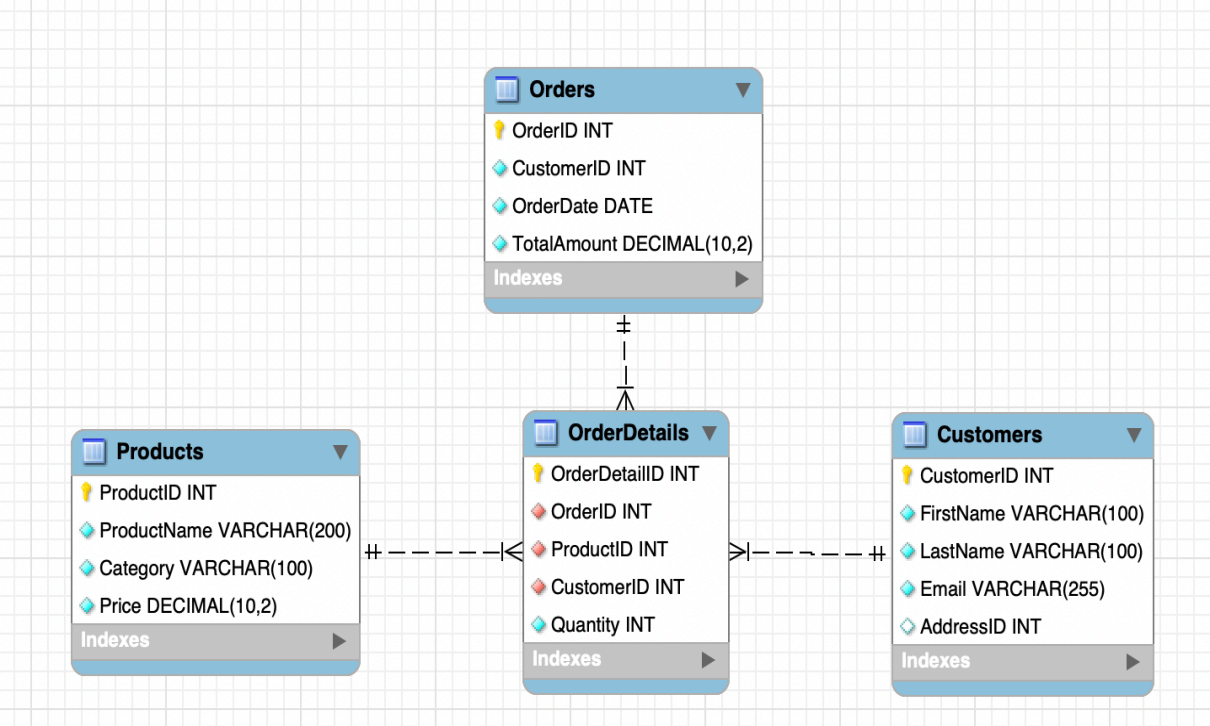


Рисунок 4.10 – Фізична схема вихідної бази даних

На рисунку 4.11 наведено фізичну схему цільової бази даних, до якої необхідно знайти відповідності а також мігрувати дані.

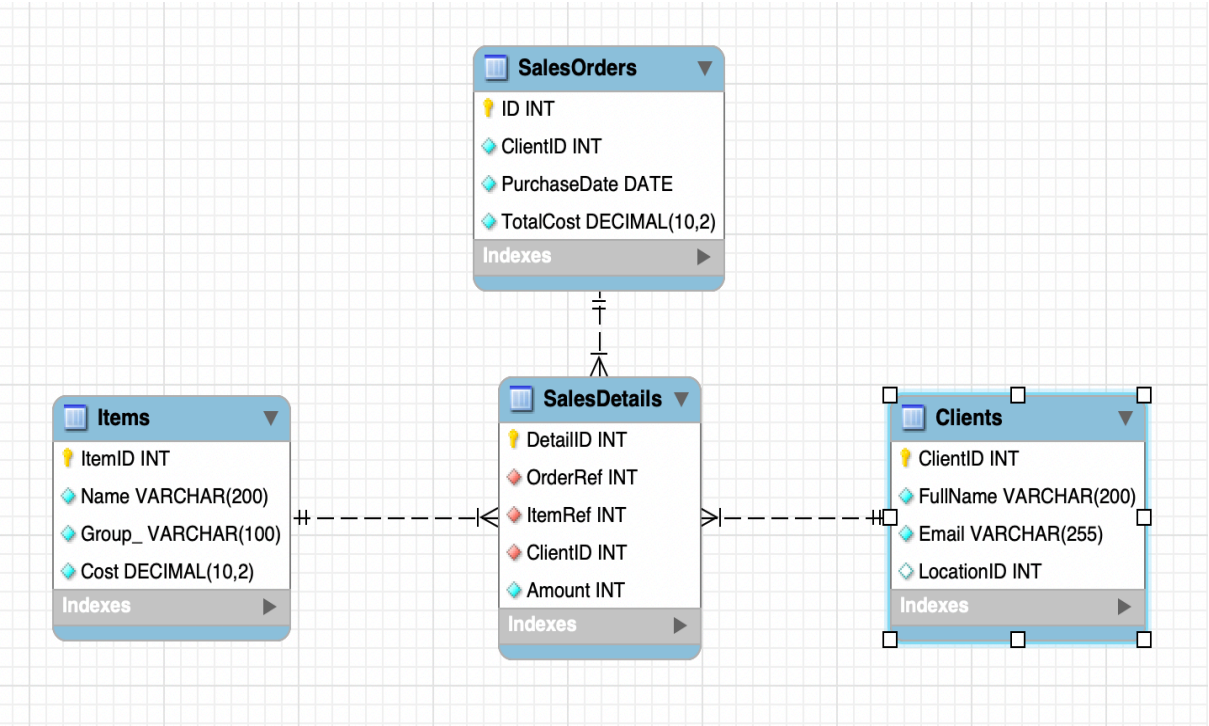


Рисунок 4.11 – Фізична схема цільової бази даних

Для створення запиту до великої мовної моделі потрібно створити промпт, який базується на прикладах. Набір таких промптів, які базується на різних прикладах, але відносяться до одного набору таблиць, наведений у таблиці 4.1.

Таблиця 4.1 – Приклади запитів до великої мовної моделі

Назва типу	Текст запиту
Просте співставлення таблиць	Є таблиця "Orders" із атрибутами "OrderID", "CustomerID", "OrderDate", "TotalAmount" у вихідній базі даних. Яка таблиця та атрибути у цільовій базі відповідають цим елементам?

Продовження таблиці 4.1

Назва типу	Текст запиту
Складне співставлення атрибутів	Є таблиця "OrderDetails" із атрибутами "OrderDetailID", "OrderID", "ProductID", "Quantity" у вихідній базі даних. Які атрибути у таблиці "SalesDetails" у цільовій базі відповідають цим елементам?
Трансформація даних	Є таблиця "Products" із атрибутами "ProductID", "ProductName", "Category", "Price" у вихідній базі даних. У цільовій базі таблиця "Items" має атрибути "ItemID", "Name", "Group", "Cost". Які відповідності між цими таблицями, якщо "Price" трансформується у "Cost" з округленням до 2 знаків?
Співставлення таблиць із об'єднанням	Є таблиця "Orders" із атрибутами "OrderID", "CustomerID", "OrderDate", "TotalAmount" та таблиця "OrderDetails" із атрибутами "OrderDetailID", "OrderID", "ProductID", "Quantity" у вихідній базі даних. У цільовій базі ці таблиці об'єднані у "SalesOrders". Які відповідності можна встановити між вихідними таблицями та цільовою таблицею?

Кінець таблиці 4.1

Назва типу	Текст запиту
Пошук зв'язків між таблицями	Є таблиця "Customers" із атрибутами "CustomerID", "FirstName", "LastName", "Email", "AddressID" у вихідній базі даних та таблиця "Clients" із атрибутами "ClientID", "FullName", "Email", "LocationID" у цільовій базі. Як співставити ці таблиці, якщо "FullName" є об'єднанням "FirstName" і "LastName"?

Власноруч запустивши ці запити до великої мовної моделі, отримуємо такі результати.

Просте співставлення таблиць із промптом: «Є таблиця "Orders" із атрибутами "OrderID", "CustomerID", "OrderDate", "TotalAmount" у вихідній базі даних. Яка таблиця та атрибути у цільовій базі відповідають цим елементам?» має результати, наведені у таблиці 4.2.

Цей запит спрямований на автоматизацію співставлення таблиць і атрибутів, використовуючи відповідність між назвами та типами даних. LLM аналізує запит і повертає потенційно відповідні таблиці та атрибути з цільової бази.

Таблиця 4.2 – Результати роботи промпту для простого співставлення таблиць.

Вихідний атрибут	Цільовий атрибут
OrderID	ID

Кінець таблиці 4.2

Вихідний атрибут	Цільовий атрибут
CustomerID	ClientID
OrderDate	PurchaseDate
TotalAmount	TotalCost

Складне співставлення атрибутів із промптом: «Є таблиця "OrderDetails" із атрибутами "OrderDetailID", "OrderID", "ProductID", "Quantity" у вихідній базі даних. Які атрибути у таблиці "SalesDetails" у цільовій базі відповідають цим елементам?» має результати, які наведені у таблиці 4.3.

Запит орієнтований на пошук відповідностей для складних структур, де можливі зміни в назвах атрибутів або форматах даних. Модель надає результати у вигляді відповідностей, наприклад, "OrderID → OrderRef".

Таблиця 4.3 – Результати роботи промпту для складного співставлення атрибутів

Вихідний атрибут	Цільовий атрибут
OrderDetailID	DetailID
OrderID	OrderRef
ProductID	ItemRef
Quantity	Amount

Трансформація даних із промптом: «Є таблиця "Products" із атрибутами "ProductID", "ProductName", "Category", "Price" у вихідній базі даних. У цільовій базі таблиця "Items" має атрибути "ItemID", "Name", "Group", "Cost". Які відповідності між цими таблицями, якщо "Price" трансформується у "Cost" з округленням до 2 знаків?» має результати, які наведені у таблиці 4.4.

Цей запит враховує додаткові вимоги до трансформації даних під час міграції, наприклад, зміни формату числових значень. LLM повертає відповідності з урахуванням трансформацій.

Таблиця 4.4 – Результати роботи промпту для трансформації даних

Вихідний атрибут	Цільовий атрибут
PorductID	ItemID
ProductName	Name
Category	Group
Price	Cost (з урахуванням до 2 знаків)

Співставлення таблиць із об'єднанням із промптом: «Є таблиця "Orders" із атрибутами "OrderID", "CustomerID", "OrderDate", "TotalAmount" та таблиця "OrderDetails" із атрибутами "OrderDetailID", "OrderID", "ProductID", "Quantity" у вихідній базі даних. У цільовій базі ці таблиці об'єднані у "SalesOrders". Які відповідності можна встановити між вихідними таблицями та цільовою таблицею?» має результати, наведені у таблиці 4.5

Мета цього запиту — визначити відповідності між вихідними таблицями та об'єднаною таблицею у цільовій базі. LLM аналізує зв'язки між атрибутами та надає пропозиції щодо їх співставлення.

Таблиця 4.5 – Результати роботи промпту для співставлення таблиць із об’єднанням

Вихідний атрибут	Цільовий атрибут
OrderID	ID
CustomerID	ClientID

Кінець таблиці 4.5

Вихідний атрибут	Цільовий атрибут
OrderDate	PurchaseDate
TotalAmount	TotalCost
ProductID	ItemRef
Quantity	Amount

Пошук зв'язків між таблицями із промптом: «Є таблиця "Customers" із атрибутами "CustomerID", "FirstName", "LastName", "Email", "AddressID" у вихідній базі даних та таблиця "Clients" із атрибутами "ClientID", "FullName", "Email", "LocationID" у цільовій базі. Як співставити ці таблиці, якщо "FullName" є об'єднанням "FirstName" і "LastName"?» має результати, які наведені у таблиці 4.6.

Запит враховує специфічні трансформації, такі як об'єднання атрибутів у цільовій таблиці. LLM повертає результати із зазначенням способу об'єднання, наприклад, "FullName → CONCAT(FirstName, ' ', LastName)".

Таблиця 4.6 – Результати роботи промпту для пошуку зв’язків між таблицями

Вихідний атрибут	Цільовий атрибут
CustomerID	ClientID
FirstName + LastName	FullName (CONCAT(FirstName, ` `, LastName))
Email	Email
AddressID	LocationID

Для порівняння результатів будемо використовувати декілька параметрів:

– точність співставлення: порівнюється кількість правильно визначених відповідностей між атрибутами та таблицями;

– час виконання: враховується час, витрачений на ручне співставлення та автоматизацію із використанням LLM;

– економія людиногодин: розраховується різниця у витратах часу між двома методами.

Результати порівняння методу ручного співставлення атрибутів і методу запиту до великої мовної моделі щодо міграції даних наведені у таблиці 4.7.

Таблиця 4.7 – Результати порівняння методів співставлення

Критерій	Ручне співставлення	Запит до LLM	Примітки
Точність співставлення	96%	92%	LLM пропустив кілька складних відповідностей
Час виконання	3 години (на 1 набір таблиць)	20 хвилин	Швидкість LLM значно вища
Людиногодини на 10 наборів таблиць	30 годин	5 годин	Економія 25 годин

Як можна побачити з таблиці, точність співставлення мовної моделі майже дорівнює точності ручного співставлення. З іншого боку, швидкість ручного співставлення дуже різниться від швидкості запиту. Більш детальні результати для наборів таблиць наведені у таблиці 4.8.

Таблиця 4.8 – Результати для набору таблиць

Таблиця	Кількість атрибутів	Час (ручний метод)	Час (LLM)	Точність (ручний метод)	Точність (LLM)
Orders	4	45 хв	5 хв	100%	100%
Customers	5	50 хв	6 хв	100%	90%
Products	4	40 хв	5 хв	100%	95%
OrderDetails	4	45 хв	7 хв	95%	90%
Загалом	17	3 години	23 хв	96%	92%

Як можна побачити в результуючій таблиці, точність різниться лише у 4%. На великих наборах даних це дуже маленька різниця, адже всі співставлення, згенеровані великою мовною моделлю, будуть перевірені додатково фахівцями з міграції.

Тобто, для набору із 10 таблиць для співставлення ручний метод займає 30 годин, а запит до великої мовної моделі щодо міграції даних займає ~5 годин. Економія = 25 годин (83% скорочення часу).

Враховуючи всі вищеперелічені дані, можна зазначити, що запит до великих мовних моделей скорочує час співставлення таблиць та атрибутів для міграції на 83%, що робить цей підхід особливо корисним для великих обсягів даних, що притаманне ІС електронної комерції [20]. Хоча LLM демонструє трохи нижчу точність у порівнянні з ручним методом, його результати можуть бути використані як базові з подальшою перевіркою вручну. Перелік позитивних сторін від використання цього методу під час міграційних процесів наведений на рисунку 4.12.

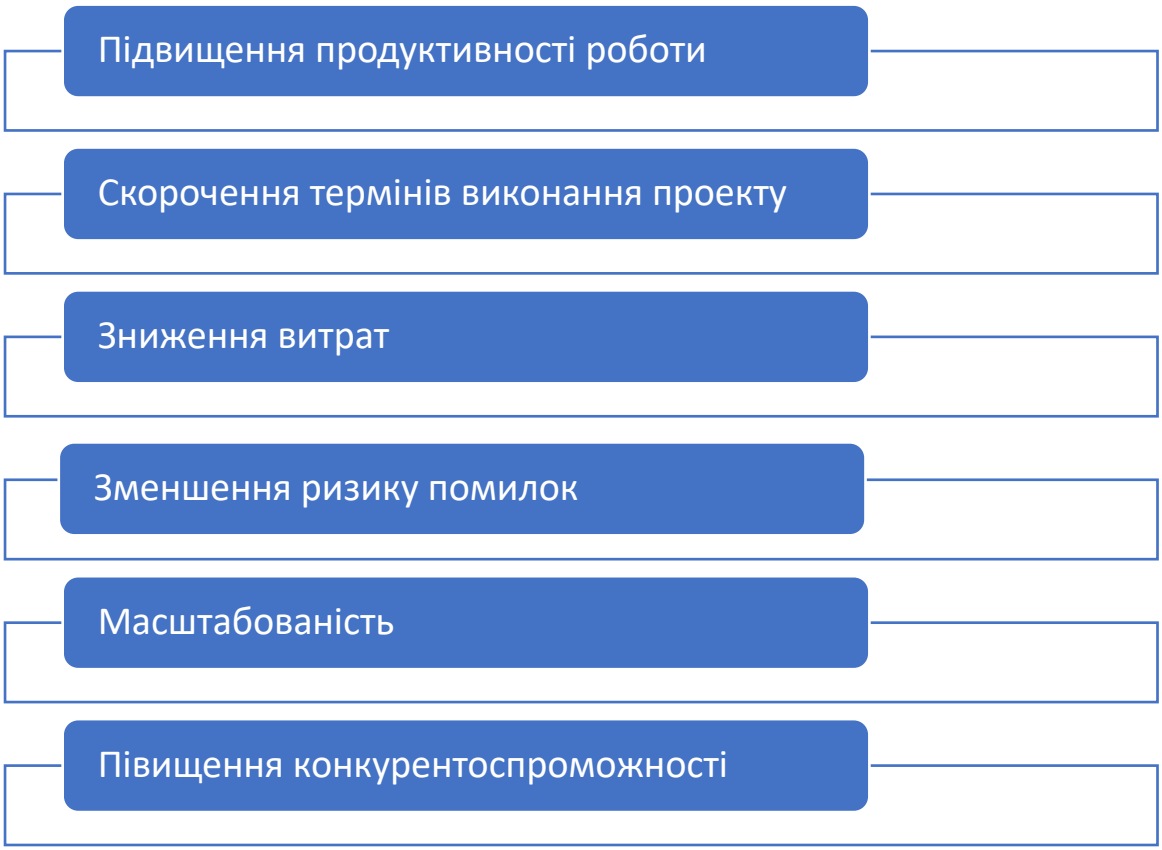


Рисунок 4.12 – Позитивні сторони використання запиту до великих мовних моделей щодо міграції даних

Використовуючи рисунок зазначений вище, можна деталізувати кожну перевагу використання.

Підвищення продуктивності роботи полягає в тому, що економія людиногодин у 25 годин (на 10 наборів таблиць) дозволяє значно підвищити продуктивність команди. Замість витрати часу на рутинну ручну роботу, спеціалісти можуть зосередитися на більш складних і стратегічно важливих завданнях, таких як:

– розробка та оптимізація процесів міграції даних;
– аналіз помилок та удосконалення системи;
– інтеграція нових функцій до інформаційних систем [21].

Скорочення термінів виконання проекту відбувається завдяки автоматизації, яка дозволяє завершувати завдання з міграції даних набагато

швидше. Якщо раніше на співставлення 10 наборів таблиць вручну витрачалося 30 годин, то за допомогою LLM це займає лише 5 годин. Це скорочення часу:

– прискорює етапи підготовки до міграції;
– зменшує загальну тривалість проекту;
– дозволяє швидше перейти до наступних етапів впровадження.

Зниження витрат полягає в економії часу, що автоматично знижує витрати на оплату праці команди. Якщо годинна ставка спеціаліста становить, наприклад, $30, то:

– ручний метод: 30 годин × $30 = $900;
– LLM-метод: 5 годин × $30 = $150;
– економія: $750 на 10 наборів таблиць.

У великих проектах, що включають сотні наборів таблиць, економія фінансових ресурсів може досягати тисяч і навіть десятків тисяч доларів.

Зменшення ризику помилок полягає в тому, що автоматизація зменшує ризик людських помилок, які можуть виникати через втому чи невірну інтерпретацію даних. Чітке формулювання запитів до LLM дозволяє отримати більш структуровані результати з меншою ймовірністю некоректних відповідностей, що:

– підвищує якість міграції даних;
– зменшує витрати на виправлення помилок у майбутньому.

Масштабованість дозволяє LLM масштабувати процес співставлення на великі обсяги даних без значного збільшення витрат часу. Наприклад:

– 100 наборів таблиць вручну = 300 годин;
– 100 наборів таблиць з LLM = 50 годин.

Це робить можливим реалізацію проектів з великим обсягом даних у коротші строки.

Підвищення конкурентоспроможності відбувається завдяки швидшій реалізації проектів та оптимізації ресурсів, що дозволяє організації:

– реалізовувати більше проектів за той самий час;
– пропонувати клієнтам конкурентні ціни на послуги;
– демонструвати інноваційність і сучасний підхід до вирішення завдань.

Підсумовуючи, можна зазначити, що використання запитів до великих мовних моделей щодо міграції даних допоможе досягти економії людиногодин завдяки автоматизації зіставлення таблиць, адже це не тільки скорочує витрати та терміни виконання, але й покращує якість роботи, підвищує задоволеність команди та дозволяє організації реалізовувати масштабніші проекти.

ВИСНОВКИ

У ході роботи було досліджено процеси міграції даних в інформаційних системах електронної комерції та реалізовано експериментальну перевірку запропонованих методів. Отримано такі результати: проведено аналіз існуючих методів міграції даних, включаючи ETL, реплікацію та ручне кодування; розроблено критерії оцінки ефективності процесів міграції, зокрема точності, швидкості виконання та впливу на бізнес-процеси; створено структуру баз даних вихідної та цільової систем для проведення експериментального дослідження; розроблено програмні модулі для автоматизації міграції, включаючи інтеграцію запитів до великих мовних моделей (LLM) для спрощення співставлення таблиць та атрибутів; спланово та проведено експеримент, що продемонстрував переваги автоматизації через LLM у скороченні часу на співставлення атрибутів та підвищенні продуктивності роботи.

Актуальність роботи полягає в тому, що існуючі процесні моделі міграції даних в інформаційних системах електронної комерції стикаються з низкою викликів, таких як забезпечення безперервності роботи систем, підтримка узгодженості даних, адаптація до нових архітектур, а також ефективне використання інструментів ETL та реплікації. Усе це ускладнюється збільшенням обсягів даних і необхідністю враховувати індивідуальні особливості інформаційних систем. Традиційні методи, які використовуються для міграції, потребують ручного визначення відповідності даних людми-експертами в предметній галузі, що призводить до значних витрат часу та збільшення бюджету проектів.

Наукова новизна полягає в тому, що було удосконалено модель міграції даних в ІС електронної комерції між базами даних шляхом знаходження відповідностей між таблицями та атрибутами вхідної та

вихідної баз даних на основі використання запитів до великих мовних моделей (LLM).

В процесі дослідження було встановлено, що автоматизація процесів міграції з використанням LLM дозволяє скоротити час на виконання завдань до 80%, зменшити кількість помилок, пов’язаних із ручним співставленням, та підвищити ефективність роботи команди. Порівняльний аналіз показав, що, хоча точність LLM трохи поступається ручному методу, часова економія та простота використання роблять цей підхід доцільним для великих обсягів даних.

Подальший напрямок досліджень пов’язаний з розробкою розширеного підходу до міграції даних, який враховував би зв’язки між таблицями в базах даних, а також забезпечував би міграцію даних для Big Data та хмарних середовищ. Також перспективним є використання LLM для автоматизації аспектів міграції, пов’язаних з аналізом трансформацій даних та підтримку гібридних рішень.

За результатами роботи представлено тези доповіді «Дослідження процесів міграції даних в інформаційній системі електронної комерції» на 29-ому Міжнародному молодіжному форумі «РАДІОЕЛЕКТРОНІКА ТА МОЛОДЬ У XXI СТОЛІТТІ».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bisbal J. Legacy systems: issues and directions / J. Bisbal, D. Lawless. B. Wu, J. Grimson // IEEE Software. – 1999. – Vol. 16, Issue 5. – P. 103–111.

2. Hara T. Database migration: a new architecture for transaction processing in broadband networks / T. Hara, M. Tsukamoto // IEEE Transactions on Knowledge and Data Engineering. – 1998. – Vol. 10, Issue 5. – P. 839 – 854.

3. Lin C.–Y. Migrating to relational systems: problems, methods and strategies // Contemporary Management Research. – 2008. – Vol. 4, No. 4. – P. 369–380.

4. Tom L., Prakash N. Migrating to the Cloud: Oracle Client/Server Modernization – Syngress Publishing, 2011. – 400 p.

5. William U., Philip N. Information Systems Transformation: ArchitectureDriven Modernization Case Studies – Morgan Kaufmann Publishers 2010. – 456 p.

6. Alam M. Migration from relational database into object oriented database / M. Alam, S. Wasan // Journal of Computer Science. – 2006. – Vol. 2, Issue 10. – P. 781– 784.

7. Best practices for data migration [Electronic resource] // IBM Global Services. URL: http://www–935.ibm.com/services/us/gts/pdf/softek–best–practices–data– migration.pdf (reference date: 10.01.2011).

8. Chester B. Data migration 101 // AIIM E–DOC. – 2006. – Vol. 20, Issue 1. – P.

9. Drumm C. QuickMig – automatic schema matching for data migration projects / C. Drumm, M. Schmitt, H.–H. Do, E. Rahm // Proceedings of 16th ACM conference on Conference on information and knowledge

management (CIKM'07). Lisboa, Portugal, November 6–8, 2007. – NY, USA, 2007. – P. 107–116.

10. Data migration test strategy: Create an effective test plan. URL: https://www.datamigrationpro.com/data-migration-go-live-strategy.

11. Maatuk A. Relational database migration: a perspective / A. Maatuk., A. Ali, N. Rossiter // Database and Experts Applications, 19th International Conference (DEXA 2008). Turin, Italy, September 1-5, 2008. Proceedings. – Springer Berlin / Heidelberg, 2008. – P. 676-683.

12. P. Russom, “Best Practices in Data Migration,” Renton, WA, US, 2006.

13. Tehreem Naeem. What is data migration. The Why, The What, and The How. URL: https://www.astera.com/type/blog/data-migrationsoftware/.

14. J. Morris, Practical Data Migration, 3rd ed. Swindon, United Kingdom: British Informatics Society Ltd, 2006.

15. Gorokhovatskyi, V.A., Vechirska, I.D., Chetverikov, G.G., Method for building of logical data transform in the problem of establishing links between the objects in intellectual telecommunication systems Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 2016, 75(18), c. 1645-1655.

16. C.-Y. Lin, “Migrating to Relational Systems: Problems, Methods, and Strategies,” Contemporary Management Research, vol. 4, no. 4, pp. 369–380, 2008.

17. P. Howard and C. Potter, “Data migration in the global 2000 - research, forecasts and survey results,” London, United Kingdom, p. 29, 2007.

18. Java SE 17 Java Database Connectivity (JDBC) – related APIs & Developer Guides//OracleDocumentation.–Electronicdata.–URL: http://docs.oracle.com/javase/17/docs/technotes/guides/jdbc/index.html.

19. Manoku E. A fact approach on data migration: an algorithm for migrating data from one database system to another [Electronic resource] / E. Manoku, G. Bakema // inconcept.com.

20. Lin C.–Y. Migrating to relational systems: problems, methods and strategies // Contemporary Management Research. – 2008. – Vol. 4, No. 4. – P. 369–380.

21. Чалий С.Ф., Синяков А.А. Розробка методу побудови нечітких запитів до реляційних баз даних для задач пошуку персональної інформації // Збірник наукових праць Харківського університету Повітряних сил. – 2008. – №3. – С. 138–140.