МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

рівень вищої освіти другий (магістерський)

Дослідження методів міграціїї даних між реляційними системами управління базами даних
(тема)

Виконав:
Студент 2 курсу, групи ІПЗм-19-2
Торбієвський О.М.
(прізвище, ініціали)

Спеціальність 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

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

Керівник доц. Мазурова О.О.
(посада, прізвище)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар
(підпис) (прізвище, ініціали)

2021 р

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 121- Інженерія програмного забезпечення
(код і повна назва спеціальності)

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

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
« ____ » _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студента Торбієвського Олександра Михайловича
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження методів міграції даних між реляційними системами управління базами даних
затверджена наказом університету від 26.03.2021 № 385
2. Термін подання роботи до екзаменаційної комісії 09 травня 2021р.
3. Вихідні дані до роботи *електронні ресурси за обраною тематикою, порівняльний аналіз існуючих методів міграції, аналіз популярності СУБД, середовища розробки Talend Open Studio, JetBrains Rider, мови T-SQL, C#, JavaScript.*
4. Перелік питань, що потрібно опрацювати в роботі *аналіз предметної області і постановка задачі, огляд методів міграції даних між РСУБД розробка програмних реалізацій для тестування різних методів міграції, проведення експерименту, формування рекомендацій.*
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів) *21 слайд презентації*

6. Консультанти розділів роботи (п.6 включається до завдання 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
Спецчастина	доц. Мазурова О.О.		12.05.21

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Аналіз проблемної області дослідження	21.01.21– 30.01.21	виконано
2	Розробка постановки задачі	8 лютого 2021р.	виконано
3	Дослідження існуючих методів міграції даних	15 лютого 2021р.	виконано
4	Планування експериментальної частини дослідження	22 лютого 2021р.	виконано
5	Проектування бази даних для дослідження	5 березня 2021р.	виконано
6	Програмна реалізація засобів для експерименту	21 березня 2021р.	виконано
7	Проведення експериментів	14 квітня 2021р.	виконано
8	Апробація результатів	28 квітня 2021р.	виконано
9	Підготовка пояснювальної записки	30 квітня 2021р.	виконано
10	Підготовка презентації та доповіді	5 травня 2021р.	виконано
11	Нормоконтроль	12 травня 2021р.	виконано
12	Рецензування	12 травня 2021р.	виконано
13	Занесення диплома в електронний архів	12 травня 2021р.	виконано
14	Попередній захист	13 травня 2021р.	виконано
15	Допуск до захисту у зав. кафедри	14 травня 2021р.	виконано

Дата видачі завдання 21 січня 2021р.

Студент _____
(підпис)

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

РЕФЕРАТ / ABSTRACT

Кваліфікаційна робота містить 113 с., 21 рис., 3 табл., 25 джерел.

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

Об’єктами дослідження виступають реляційні системи управління базами даних та методи міграцій даних у контексті реляційних баз даних

Метою дослідження є знаходження оптимальних підходів до процесу міграцій даних між реляційними системами управління базами даних різних виробників та виведення рекомендацій з їх проведення.

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.

EXTRACT, LOAD, DATA MIGRATION, MIGRATION PLAN, RELATIVE DBMS, REPLICATION, TRANSFORMATION, ETL, SQL.

The objects of the study are relational database management systems and data migration methods in the context of relational databases

The aim of the study is to find optimal approaches to the process of data migration between relational database management systems of different manufacturers and derive recommendations for their implementation.

The research to be conducted is based on the results of the analysis of existing approaches to data migration, the possibility of their application between different database management systems.

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

Я, Торбієвський Олександр Михайлович, студент групи ІПЗм-19-2, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження методів міграції даних між реляційними системами управління базами даних», що буде представлена в екзаменаційну комісію для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Всі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів

ЗМІСТ

ВСТУП

Сучасний світ майже у всіх сферах життя використовує комп’ютери. Розвиток інформаційних технологій нерозривно пов'язаний зі збільшенням кількості збережених даних. Тому очевидна необхідність розвитку спеціалізованих інструментів. Майже у кожній сфері життєдіяльності існують свої варіанти спеціалізованого програмного забезпечення, кожен з них потребує власного сховища даних, якими, в свою чергу, виступають реляційні системи управління базами даних. Інструментарій сучасних реляційних систем управління базами даних постійно вдосконалюється і розвивається, з'являються нові сервіси та програмні комплекси.

На сьогоднішній день реляційна модель даних є основною при побудові баз даних. Так відбувається, тому що організація інформації у вигляді таблиць інтуїтивно зрозуміла користувачу. Ця особливість привела до появи безлічі реляційних систем управління базами даних, характеристики яких істотно розрізняються. Підбір найбільш підходящою системи управління базами даних став нетривіальним завданням.

З часом, кількість та об’єм даних тільки зростають, що створює поштовх до постійного вдосконалення цих сховищ та їх модернізації. На перший погляд створюється враження, що проблеми можуть виникати тільки на етапі первинного проектування, але на практиці це не так - основною проблемою є постійна підтримка та “еволюція” таких сховищ, що, зрештою, може призвести до виникнення необхідності у повній міграції програмної системи на нову версію сховища, або, у окремих випадках, на зовсім інше сховище. Саме з цієї причини виникає потреба у міграції даних.

Міграція даних - процес перенесення великих об’ємів даних із одного місця в інше. Як приклад міграції даних, можна взяти процес перенесення файлів із переносного накопичувача на стаціонарний, але, у випадку сховищ, що використовуються бізнесом, процес набуває набагато більших масштабів. Бізнес

може вимагати як і відносно простої зміни схеми сховища, так і повного переходу на інший тип сховища, як наслідок - кожна, з цих міграцій, несе відповідні ризики, які повинні бути мінімальними, та непомітними для кінцевого користувача.

Міграції даних використовуються бізнесом для підвищення ефективності його ведення та безпеки. Модернізація технологій та відмова від застарілих систем - невідворотна складова сучасного бізнесу. На даний момент не існує єдиного стандартизованого підходу до процедури міграцій даних. Кожен випадок є унікальним та потребує розробки власного плану та, зазвичай, спеціальних інструментів, що вирішують ту чи іншу прикладну задачу.

В результаті кваліфікаційної роботи було проведено аналіз існуючих методів міграції на основі праць вітчизняних і зарубіжних вчених (див. додаток А), можливість їх застосування у реляційнх СУБД та доцільність їх застосувань у різних ситуаціях, розроблено систему критеріїв оцінювання якості та сплановано експериментальне дослідження. Розроблено інструмент для проведення міграцій та оцінки якості даних після їх проведення, побудовано плани міграції, схеми бази даних для проведення есперименту, сформульовано рекомендації щодо проведення міграцій, їх планування та застосування.

Отримані результати можуть бути застосовані при створенні планів міграцій та їх проведенні.

Робота перевірена на унікальність тексту в мережі інтернет та базі ХНУРЕ та на відповідність вимогам оформлення (див. додаток Б та В).

За результатами кваліфікаційної роботи магістра було розроблено презентацію (див. додаток В). За результатами роботи створено тези доповіді на участь у науково-технічній конференції «Сучасні напрямки розвитку автоматизації, транспортних систем, технічних та комп’ютерних наук» (див. додаток Г). Лістинг коду наведено в додатку Д.

1 АНАЛІЗ ПРОБЛЕМНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз проблемної області дослідження

Розробники різноманітних програмних засобів та додатків постійно стикаються з проблемами, що виникають через збільшення об’ємів даних та зміни у їх 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.

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 штовхають 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 озера даних або іншої більш гнучкої системи.

– одне джерело істини – переміщення всіх даних в одне місце, доступне для всіх підсистем.

Технічно, міграція до іншого сховища даних складається з:

– міграція 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;

– міграція 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;

– міграція 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 невидима 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.

План міграції даних (див. рис. 1.1) повинен враховувати наступні фактори:

– аудит та дослідження даних та їх структури - перед міграцією вихідні дані повинні пройти повний аудит. Якщо ігнорувати цей крок, можуть з’явитися несподівані проблеми.

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

– журналювання - відстеження та звітування про якість даних є важливим, оскільки це дозволяє краще зрозуміти цілісність та якість даних.

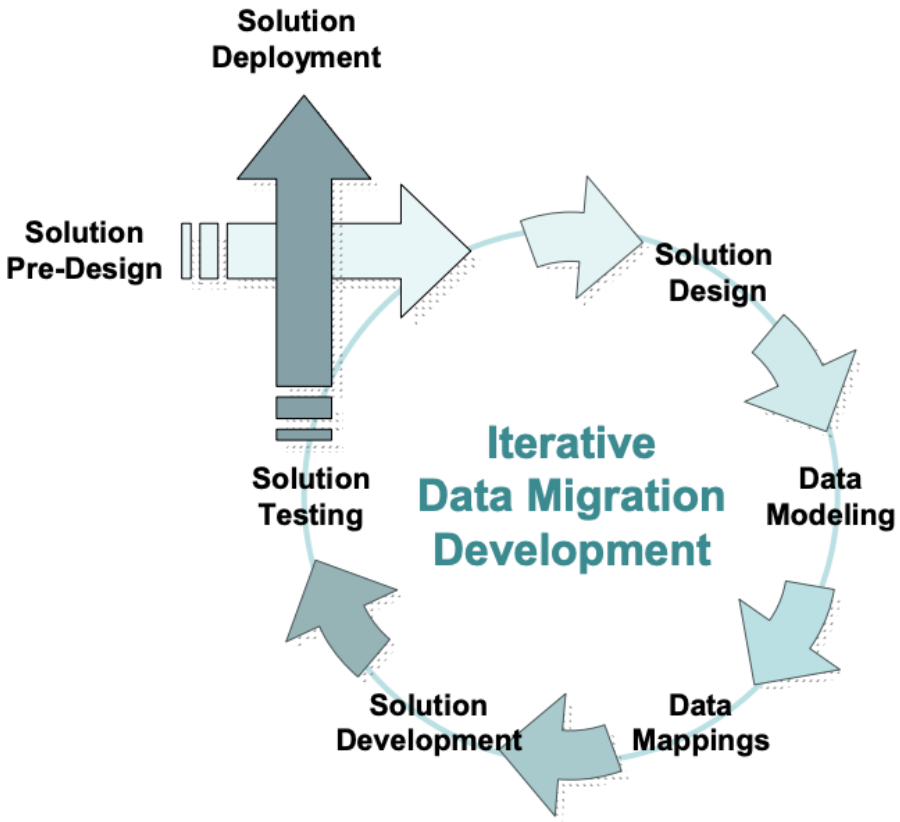

Рисунок 1.1 - Типова схема моделювання процесу міграції даних

На додаток, добре спланований процес міграції повинен включати процеси інсталяції та конфігурації нової СУБД та прикладних до неї систем [6].

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 [7]:

– витяг (extract) - вилучення даних із вхідної СУБД;

– трансформація (transform) - використовується для застосування ряду правил або функцій на вхідні дані (вилучені дані), такі як об'єднання, агрегування або функції фільтру, щоб зробити вилучені дані у придатній для використання формі для переходу до цільової бази даних;

– завантаження (load) - процес завантаження даних у цільову СУБД.

Іноді, розробникам доводиться переходити на зовсім нову СУБД, наприклад, з MSSQL на MySQL, або, з Oracle на MSSQL. Це одна з найскладніших міграцій, оскільки ми маємо не тільки перенести дані, але й перенести тригери, процедури. Міграція тригерів та збережених процедур досить складна через зміни синтаксису, як, наприклад, якщо ви переходите із Oracle на MSSQL, в Oracle збережені процедури будуть записані мовою PL / SQL, тоді як у SQL Server - на T-SQL.

Існує дві найбільш поширені імплементації підходу ETL:

– експорт даних до файлів CSV/JSON, їх завантаження на сервер нової бази даних та подальший їх імпорт до нової СУБД;

– пряме перенесення даних із однієї СУБД до іншої без використання проміжних файлів, тобто, виконуємо запит на вибірку у поточній СУБД, транформуємо дані та виконуємо запит на запис у цільовій СУБД.

Також, існує інша проблема - міграція даних у вже працюючих “живих” системах. У цьому випадку, міграція стає ще більш складною, бо користувачі працюють з системою у реальному часі. Зазвичай, у випадку реляційних баз даних, такі проблеми вирішуються за допомогою безперервної синхронізації атомарних транзакцій (підтримки узгодженості) між поточною та цільовою СУБД [8]. Існує декілька варіантів вирішення цієї проблеми.

Можна організувати синхронізацію за допомогою логування транзакцій. Кожного разу, коли у вхідній СУБД проводиться транзакція - вона записується у спеціальне сховище, разом із її докладним описом (списком операцій, які вона містить), потім, у деякий момент часу, ці транзакції можуть бути проведені у цільовій СУБД. Деякі реляційні СУБД логують транзакції автоматично, що

прибирає потребу в розробці власних механізмів логування. Цей підхід може бути використаний при проведенні міграцій у невеликих системах, де кількість операцій, які проводяться користувачами, не є дуже великою.

Іншим варіантом є проведення транзакцій у всіх СУБД одночасно з їх появою, тобто, після проведення транзакції у поточній СУБД, вона транслюється у діалект цільової СУБД та виконується там. Такий підхід ще можна назвати “підтриманням постійної транзакційної узгодженості”. У цьому випадку дуже важливо пам’ятати про рівні ізоляції вищевказаних транзакцій, бо вони могли виконуватися конкурентно.

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. Однак ними можна керувати та пом'якшувати за допомогою чітко визначеної стратегії та плану перевірки.

1.2 Постановка задачі

Метою дослідження є знаходження оптимальних підходів до процесу міграцій даних між реляційними системами управління базами даних різних виробників та виведення рекомендацій з їх проведення.

Об’єктами дослідження виступають різноманітні реляційні системи управління базами даних, підходи до міграцій даних та особливості використання цих підходів у контексті реляційних баз даних

Таким чином, під час дослідження необхідно вирішити наступні задачі:

– провести аналіз та вибір СУБД для дослідження міграції;

– провести аналіз підходів та методів міграції між обраними СУБД, та обрати певні для дослідження;

– спланувати експериментальне дослідження процесів міграції (розробити систему критеріїв для оцінки якості міграції; розробити схему та інші об'єкти БД для проведення міграцій);

– розробити плани міграції між відповідними СУБД на базі обраних підходів та реалізувати їх;

– провести експериментальне дослідження міграцій, оцінити їх якість та виробити рекомендації стосовно міграцій.

2 ОПИС ПРИЙНЯТИХ ПРОЕКТНИХ РІШЕНЬ

2.1 Аналіз та визначення проблем під час міграцій баз даних

Міграції даних між різноманітними системами управління базами даних є достатньо нестандартною задачею, бо навіть на даний момент не існує єдиного підходу до їх проведення, що призводить до появи безлічі різноманітних підходів та практик.

Існує такий термін, як “гравітація даних” - це термін, який створив Дейв Маккрорі, віце-президент з інженерних питань GE Digital, він був вперше вжитий в публікації в його блозі в 2010 році [9]. Цей термін є аналогією, що стосується взаємозв'язку між даними та додатками - вони притягуються один до одного, аналогічно до закону тяжіння. Набори даних стають більшими, що ускладнює їх переміщення (з точки зору аналогії, вони набувають більшої “сили тяжіння”). Це призводить до того, що дані залишаються нерухомими, тоді як додатки, які «притягуються» до даних, переміщуються.

Ця концепція існує вже деякий час, але сама проблема стає все більш суттєвою. “Гравітація даних” описує такі проблеми:

- як дані взаємодіють з іншими даними по мірі збільшення їх кількості;
- як дані інтегруються у бізнес-процеси;
- як дані змінюються з часом.

Для зменшення кількості проблем, що виникають внаслідок, Гартнер рекомендує "розплутувати" дані та програми, розбиваючи їх на різні шари, такий підхід є засобом для подолання сили тяжіння даних. Тому, на старті проекту важливо виділяти час на проектування шару управління даними та зменшення кількості бізнес-логіки, що у ньому присутня, саме так можна зменшити кількість проблем, що з'явиться у майбутньому при виникненні потреби у міграції до інших сховищ.

Основною проблемою є те, що поява нових додатків у системі, у більшості випадків, додає нову логіку до шару управління даними, що ускладнює повторне

використання даних. Тому, дуже важливо проектувати додатки так, щоб дані, які вони використовують, можно було використовувати і у інших додатках. Також, при появі нових вимог або змін, дуже важливо не забувати про вищевказані проблеми, бо, за час “еволюції” системи, розробники можуть про них забути і починають з’являються так звані “витоки контекстів” (змішення бізнес-процесів із управлінням даними). Тому, архітектури додатків, даних та бізнес–процеси повинні реагувати один на одного, але часто одна з цих груп стає занадто складною для внесення змін. І, хоча, на деякий час, поява такого “витоку контексту” може бути вимушеним кроком, це є технічною заборгованістю, яка повинна бути погашена.

Переміщення важливих або конфіденційних даних та виведення із експлуатації застарілих систем може поставити зацікавлені сторони у скрутне становище. Тому, обов’язковою є наявність чіткого плану. Нині можна знайти численні зразки планів та контрольних списків міграції даних.

Наприклад, Data Migration Pro, спільнота фахівців з міграції даних, має вичерпний контрольний перелік [10], який окреслює 7–фазний процес:

– преміграційне планування – включає в себе окреслення залучених сторін та ресурсів, обмежень з точки зору безпеки, цілей міграції;

– ініціація проекту - на цьому етапі важливо формалізувати спосіб інформування кожної зацікавленої сторони. Важливо створити загальну політику заздалегідь, з кожним окремим учасником процесу;

– аналіз предметної області - описання структури, значення, змісту та контексту даних. Встановлення надійного процесу управління правилами якості даних та інформування бізнесу про цілі проекту, включаючи виведення з експлуатації застарілих систем;

– проектування рішення - власне проектування логіки підходу до міграції та її кроків;

– збірка та тестування - кодування логіки міграції та її тестування у тестовому середовищі;

– виконання та тестування - виконання міграції на живому середовищі з подальшим моніторингом та профілюванням перенесених даних, перевірка функціоналу, що з ними зв’язаний;

– зняття та моніторинг - моніторинг подальшої реакції та поведінки системи, виведення з експлуатації старих систем.

Існує багато факторів, що визначають правильну техніку міграції [11], такі як доступні ресурси, обсяг даних та їх чутливість, бізнес-вимоги. Отже вибраний метод міграції, який буде використовуватися, повинен поєднувати надійність, ефективність та здійснювати мінімальний впливу на кінцевих користувачів та бізнес-процеси.

Незважаючи на те, що з проблемою міграції даних стикаються вже протягом кількох десятиліть, спеціалісти, що їх проводять, кожен раз стикаються з одними й тими ж проблемами [12].

Комунікація. Незалежно від кількості даних, що переміщується, є кілька сторін, що зацікавлені та залучені до процесу. При плануванні міграції потрібно знайти цих людей та підтримувати з ними контакт. В першу чергу потрібно узгодити цілі процесу та, після, постійно підтримувати контакт з вказаними вище сторонами. Регулярна комунікація допомагає в уникненні комунікаційних проблем та непорозумінь.

Досвід. Часто, керівників проектів дуже хочуть зекономити при наймі або призначенні персоналу на проект з міграції, але даний процес включає в себе, серед іншого, розуміння складності профілювання даних, їх очищення та підтримку вимог безпеки. Економія коштів на дані ресурси може обійтись дуже великою ціною в довгостроковій перспективі. Процес міграції даних пов’язаний з великою кількістю ризиків та складностей. Наявність досвідченого спеціаліста допоможе зменшити часові витрати на планування та виконання, а також допоможе в уникненні цих самих додаткових ризиків.

Планування. Велика частина роботи з міграції даних передбачає ретельне планування проекту. Потрібно зібрати вимоги, узгодити показники успіху, спланувати зіставлення схем, відображень даних та створити плани резервного

копіювання та відновлення. Кожен із цих кроків займає значний час. Наявність плану міграції зменшує вірогідність появи проблем у декілька разів, бо чіткий структурований план допомагає бачити масштаб міграції та допомагає уникнути дуже багатьох проблем, що виникають при виконанні у хаосі.

Розуміння предметної області та залежностей між об’єктами. Дуже часто, спеціалісти, що проводять міграцію, не мають повного розуміння предметної області набору даних, з яким вони працюють. Це призводить до неможливості оцінювання якості перенесених даних та, як наслідок, працездатності програмного забезпечення, що з ними взаємодіє. Отже, план міграції повинен включати оцінку даних, дослідження предметної області та визначення залежностей між об’єктами. Також слід звернути особливу увагу на залежності програмних засобів від даних, які зачіпає міграція, це також може призвести до проблем. Складний продукт може містити дуже багато різних об’єктів даних, що надходять із сотні різних додатків.

Резервне копіювання та відновлення. Іноді, при виконанні міграцій, спеціалісти забувають про дуже важливі задачі із створення резервних копій на кожному етапі. Це може здатися не дуже важливим, бо помилки можна виправити вже після міграції набору даних, але це не завжди так, бо міграція може відбуватися у “живому” середовищі, тобто на працюючому програмному засобі. Кращий спосіб уникнення затримок та проблем із якістю даних це своєчасне створення планів відновлення стану на різних етапи міграції. Це передбачає проведення перевірок якості, налаштування резервних копій та інструкцій з їх відновлення. Це займає більше часу на етапі планування, але, знову ж, допомагає уникнути багатьох проблем у довгостроковій перспективі.

Очищення та профілювання. При виконанні міграцій, іноді здається, що найлегшим шляхом буде просто перемістити дані та очистити їх, коли вони перенесені до цільового сховища. Міграція, без виконання вищевказаних дій, може призвести до, якнайменше - збереженню існуючих проблем, якнайбільше - повному спотворенню даних у цільовому сховищі. Перш ніж переміщувати дані, слід витратити час на їх профілювання та очистку. Профілювання - це ретельна

перевірка наявних даних. Це допоможе зрозуміти, чи є порожні чи нульові значення, чи дані є унікальними, чи значення потрапляють у очікуємий діапазон, тощо. Наступним кроком є очищення даних, що включає в себе видалення сторонніх даних, заповнення відсутніх даних та їх нормалізацію.

2.2 Аналіз методів міграції даних

Типовий процес ETL включає в себе збір та вдосконалення різних типів даних, а також їх доставку до цільового сховища даних. ETL також дозволяє переносити дані між різними джерелами, напрямками та інструментами аналізу. Як результат, процес ETL має вирішальну роль у процесі бізнес-аналітики та виконанні багатьох стратегій управління даними.

Процес ETL складається з трьох етапів і дозволяє інтегрувати дані від вихідного сховища до цільового. Ці етапи - вилучення даних, перетворення даних та завантаження даних.

Вилучення даних. Мало які з підприємств покладається на єдиний тип даних або систему. Більшість управляє даними з різних джерел і використовує ряд інструментів аналізу даних та отримання бізнес-аналітики. Щоб скласти таку складну стратегію використання даних, важливо мати можливість вільного переміщувати їх між системами та додатками.

Перш ніж дані можна буде перемістити до цільового сховища, спочатку їх потрібно вилучити з вихідного. На першому кроці процесу ETL структуровані та неструктуровані дані імпортуються та консолідуються в єдине проміжне сховище.

Незважаючи на те, що це можна зробити вручну, вилучення даних, кодоване вручну, може забирати багато часу та, у більшості випадків, створює велику кількість помилок. Інструменти ETL автоматизують процес вилучення та дозволяють створити більш ефективний та надійний робочий процес.

Трансформація. На цьому етапі процесу ETL повинні застосовуватися правила та норми, які забезпечують якість даних та їх доступність. Процес перетворення даних складається з декількох підпроцесів:

– очищення - усунення невідповідностей та відсутніх значення в даних;

– стандартизація - застосування загальних правила форматування до набору даних;

– дедуплікація - видалення зайвих або дублікуючих даних;

– перевірка - видалення непридатних даних та виявлення аномалій;

– сортування - впорядкування даних за типом;

– інші завдання - застосування інших правил для покращення якості даних.

Трансформація, як правило, вважається найважливішою частиною процесу ETL. Вона покращує цілісність даних і допомагає гарантувати, що дані будуть надходити до цільового сховища повністю сумісними та готовими до використання.

Завантаження. Завершальним етапом процесу ETL є завантаження трансформованих даних у цільове сховище. Дані можна завантажувати відразу (повне завантаження) або через заплановані інтервали (часткове завантаження).

Повне завантаження - все, що надходить із конвеєра трансформації, створюється у якості нових, унікальних записів у цільовому сховищі даних. Зазвичай, повне завантаження може дуже швидко ускладнити підтримку. бо кожного разу створюються набори даних, розмір яких зростає в геометричній прогресії.

Часткове завантаження - менш комплексний та більш керований підхід, Він полягає в тому, що дані завантажуються частинами та порівнюються з уже наявними, що виключає можливість появи дублікатів та гарантує якість, унікальність інформації.

Реплікація. Реплікація - це підтримка двох і більше ідентичних копій (реплік) даних на різних вузлах СУБД. Репліка може включати всю базу даних (повна реплікація), одне або кілька взаємопов'язаних відносин або їх фрагментів.

Зазвичай, реплікація між двома ідентичними серверами баз даних виконується або в двійковому режимі, або за допомогою запитів між ведучим вузлом (він же видавець) і веденим (підписник). Мета реплікації - надати в реальному часі копію головної бази даних. При цьому, дані постійно передаються від ведучого вузла до веденого, тобто від активного до пасивного, бо зазвичай реплікація виконується тільки в одну сторону. Але є і винятки, у яких можна налаштувати реплікацію між двома сховищами в обидві сторони, щоб дані передавалися від веденого до ведучого вузла в конфігурації “активний-активний”. Все це, в тому числі і каскадна реплікація, зазвичай можливо тільки між ідентичними серверами баз даних, а вибір конфігурації “активний-активний” або “активний-пасивний” залежить від потреб та доступності таких можливостей у вихідній конфігурації або рішеннях, що для цього використовуються.

Описана конфігурація можлива і між різними серверами баз даних, такий підхід називається перехресною реплікацією. СУБД можна налаштувати для прийому реплікованих даних з СУБД іншого типу, але це можливо забезпечити тільки при використанні сторонніх програмних рішень, сховища різного типу зв'язати безпосередньо неможливо, бо вони можуть мати як різницю у типах даних, так і у протоколах їх передачі. Перехресна реплікація між різними СУБД зазвичай потрібна для одноразової міграції з одного типу сховища на інший.

Ручне кодування. Інтеграція даних за допомогою самописних рішень може спочатку здатися швидким, недорогим способом побудови систем міграції даних. Але слід враховувати значні недоліки. Під “ручним кодуванням” розуміється написання повного комплексного рішення для міграції даних з нуля, не вдаючись до використання існуючих інструментів.

Сама сутність рішень написаних для конкретної міграції полягає у розробці інструментів, що виконують міграцію та стежать за її виконанням. У більшості випадків це виглядає як варіація ETL, та, зрештою, дуже добре підходить для маленьких організацій та проектів з малими об’ємами даних.

Доцільність розробки самописного рішення дуже залежить від передумов його використання, бо, наприклад, при міграції великих наборів даних, внесення

змін до компексу логіки власної розробки может бути набагато дорожчим за використання існуючих засобів з відкритим вихідним кодом, але, з іншою сторони, існуючі інструменти можуть не підтримувати цільове сховище або не мати потрібного функціоналу.

2.3 Вибір методів міграції для дослідження

Згідно з опитуванням TDWI [11], 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. 2.1).

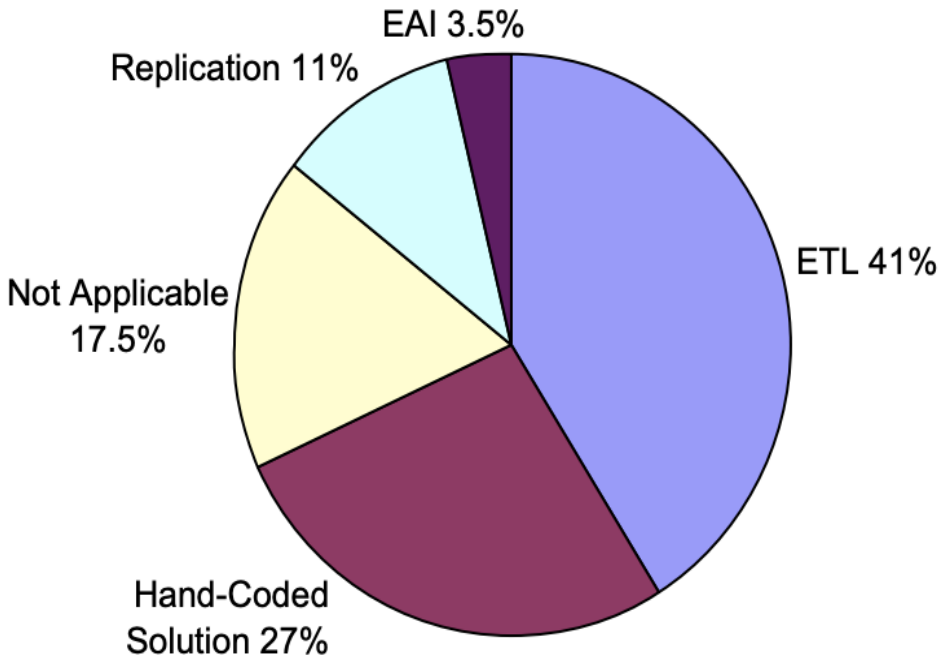

Рисунок 2.1 - Результати опитування TDWI, щодо популярності підходів до міграції даних

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.

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. 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.

2.4 Аналіз та вибір СУБД для дослідження міграції

Незважаючи на те, що всі системи управління базами даних виконують одну і ту ж основну задачу (тобто дають можливість користувачам створювати, редагувати і отримувати доступ до інформації, що зберігається в базах даних), сам процес виконання цього завдання варіюється в широких межах. Крім того, функції і можливості кожної СУБД можуть істотно відрізнятися. Різні СУБД документовані по-різному. По-різному надається і технічна підтримка.

При порівнянні різних популярних баз даних, слід враховувати, чи зручна для користувача і масштабируема дана конкретна СУБД, а також переконатися, що вона буде добре інтегруватися з іншими продуктами, які вже використовуються [13]. Крім того, під час вибору слід взяти до уваги вартість системи і підтримки, що надається розробником.

Якщо мова йде про вибір СУБД для підприємства, то слід взяти до уваги можливість СУБД «рости» разом з розвитком організації. Малому бізнесу можуть знадобитися тільки базові функції і можливості, а також невелика кількість інформації, що розміщується в базі даних, але вимоги можуть істотно зростати з плином часу, а перехід на іншу СУБД може стати проблемою.

Існує безліч реляційних СУБД, але особливою популярністю користуються далеко не всі з них.

Oracle Database є вибором багатьох організацій, бо вона була одним с перших комерційних продуктів у сфері управління даними, а можливості, які вона пропонує, є дуже привабливими для розробників. Архітектура даної СУБД гарантує майже 100% час доступу, та те, що дані завжи будуть у безпеці. Також Oracle Database має безліч функціональних можливостей, які не підтримуються іншими СУБД, це, наприклад, Grid Infrastructure, ASM або Data Guard. Варто також виділити відточеність функціоналу управління географічними даних, підтримку резервного копіювання за допомогою інструмента RMAN та величезну

кількість інструментів інтеграції, що дозволяють розширити функціональні можливості відповідно до потреб бізнесу.

MySQL - одна з найпопулярніших баз даних для веб-додатків. Фактично, є стандартом для веб-серверів, які працюють під управлінням операційної системи Linux. Існують спеціальні платні версії, призначені для комерційного використання. У безкоштовної версії найбільший фокус робиться на швидкість і надійність, а не на повноту функціоналу, що може стати і перевагою і недоліком - в залежності від області застосування. Ця СУБД дозволяє вибирати різні рушії для системи зберігання таблиць, вони дозволяють змінювати функціонал і виконувати обробку даних, що зберігаються в різних типах таблиць. Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Завдяки відкритій архітектурі і GPL-ліцензуванню, в СУБД MySQL постійно з'являються нові типи таблиць. Вона також має простий у використанні інтерфейс, пакетні команди, які дозволяють зручно обробляти величезні обсяги даних. Система неймовірно надійна і не прагне до поглинення всіх доступні апаратних ресурси.

Microsoft SQL Server - це рішення СУБД, що розроблене та підтримується корпорацією Майкрософт. Перша версія була випущена в 1989 році, SQL Server тепер доступний у багатьох різних версіях з різними наборами функцій, включаючи версії Enterprise, Standard та Express. Дана СУБД забезпечує високу масштабованість, оскільки її можна використовувати як для невеликих, так і для великих додатків, що обумовлено тим, що її пропускна здібність дозволяє обробляти мільйони транзакцій на день. Це забезпечує кращу продуктивність та високу швидкість під час роботи програмного додатку із сховищем даних.

У 2020 році компанією Stack Exchange було проведено дуже велике опитування серед працівників сфери інформаційних технологій [14], більш ніж 65 тисяч спеціалістів стали його респондентами. Дане опитування торкалося майже усіх аспектів розробки програмного забезпечення та його проектування, одною з його частин є питання про частоту використання реляційних СУБД (див. рис. 2.2).

Також існує рейтинг DB-Engines [15], що представляє собою перелік багатьох СУБД, включаючи нереляційні. На рисунку 2.3 наведений рейтинг станом на березень 2021 року. Даний рейтинг - це перелік систем управління базами даних, класифікований за їхньою популярністю, що розраховується на основі таких критеріїв:

– кількість результатів у видачі пошукових систем Google та Bing;
– частота зв'язаних пошукових запитів у Google Trends;
– частота технічних дискусій на IT Stack Overflow та DBA Stack Exchange;
– кількість пропозицій про роботу, в яких згадується СУБД;
– кількість профілів у професійних мережах (LinkedIn);
– кількість твітів у Twitter у яких згадується система.

Значення обчислюються шляхом стандартизації та усереднення окремих параметрів.

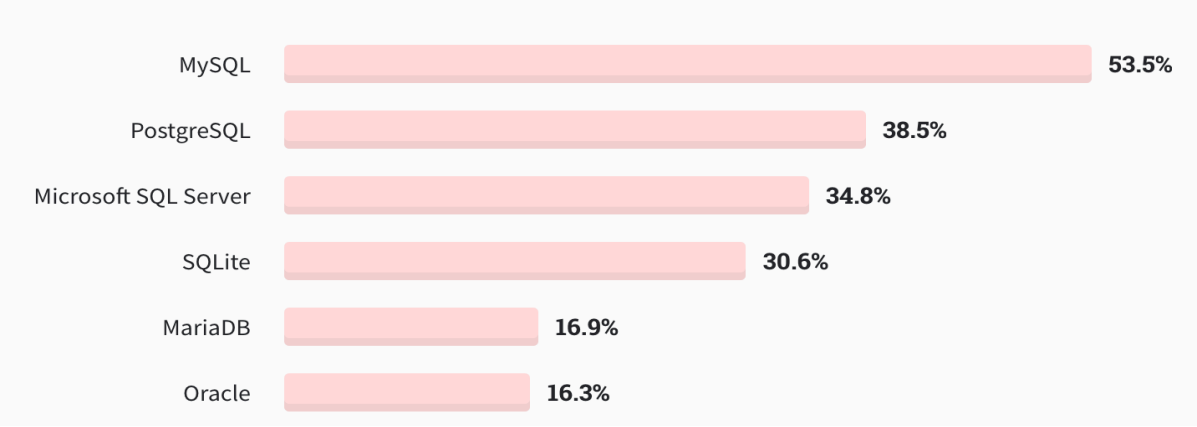

Рисунок 2.2 – Результати опитування Stack Exchange про частоту використання СУБД

Rank			DBMS	Database Model	Score		
Mar 2021	Feb 2021	Mar 2020			Mar 2021	Feb 2021	Mar 2020
1.	1.	1.	Oracle	Relational, Multi-model	1321.73	+5.06	-18.91
2.	2.	2.	MySQL	Relational, Multi-model	1254.83	+11.46	-4.90
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	1015.30	-7.63	-82.55
4.	4.	4.	PostgreSQL	Relational, Multi-model	549.29	-1.67	+35.37
5.	5.	5.	IBM Db2	Relational, Multi-model	156.01	-1.60	-6.55
6.	6.	↑ 7.	SQLite	Relational	122.64	-0.53	+0.69

Рисунку 2.3 – Рейтинг DB-Engines станом на березень 2021 року.

Для проведення експериментального дослідження нам потрібно обрати декілька набільш поширених СУБД, для цього візьмемо 6 найпопулярніших СУБД за версіями StackOverflow та DB-Engines.com та приведемо результати опитувань до коефіцієнтів популярності за допомогою принципу нормування за максимальним та мінімальним значенням. Слід зазначити, що СУБД SQLite не буде включена до порівняння, бо вона, зазвичай, використовується у вбудованих системах або мобільних додатках для зберігання невеликої кількості даних. У якості еталонного значення для результатів із StackOverflow візьмемо MySQL, для результатів DB-Engines – Oracle (див. табл. 2.1).

Таблиця 2.1 – Звідна таблиця популярності СУБД

	StackOverflow	DB-Engines	Результат сгортки
Oracle Database	0,30	1	1,3
MySQL	1	0,94	1,94
Microsoft SQL Server	0,65	0,76	1,41
MariaDB	0,31	0	0,31
PostgreSQL	0,71	0,41	1,12
IBM DB2	0	0,11	0,11

Застосуємо лінійну адитивну сгортку без вагових коефіцієнтів для отримання оптимального вибору:

$$Z^* = \max \sum_{j=1}^{n} Z_j$$

де Z_j – нормований коеффіцієнт поплуярності окремої СУБД.

Отже, виходячи з аналізу поплуярності, можна зробити висновок, що двома найпопулярнішими СУБД є MySQL та Microsoft SQL Server. Саме ці СУБД будуть використані під час поведення експериментального дослідження.

2.5 Планування експериментального дослідження

Суть експериментального дослідження полягає в застосуванні різних підходів до міграцій у тестовому середовищі на спроектованій схемі бази даних, яка містить дані та різноманітні об’єкти (функції, представлення).

Для проведення експерименту знадобиться три сервера, два з яких будуть знаходитися в одній локальній мережі, а інший – віддалений, розташований на хостингу. На всіх серверах, у якості операційної системи, буде використовуватися Ubuntu Server 20.04 – дистрибутив Linux, що використовується багатьма організаціями по всьому світу для запуску, підтримки та обслуговування систем управління базами даних. На кожному з цих серверів будуть запущені екземпляр СУБД та необхідні, для окремого типу міграції, прикладні інструменти.

При проведенні експерименту будуть проведені міграції трьох типів:

– ETL з застосуванням спецілізованого програмного комплексу;

– реплікація з використанням спеціалізованого програмного комплексу;

– застосування самописного рішення.

У якості СУБД будуть використані MySQL та Microsoft SQL Server у різних комбінаціях MySQL до Microsoft SQL Server та у зовортньому порядку.

Для проведення кожного з тестів необхідно мати встановлену вихідну СУБД з розгорнутою базою даних у повному обсязі на одному з вказаних вище серверів, та вихідну СУБД у чистому вигляді, тобто наступними кроками є:

– створення бази даних у вхідній СУБД;

– наповнення бази даних даними та створення об’єктів (представлень, функцій, тощо);

– власне проведення міграцій

– перевірка результатів за критеріями.

2.5.1 Вибір критеріїв з оцінювання якості міграції

Отже, дуже важливим аспектом процесу міграції є якість отриманих у цільовому сховищі даних. Для вимірювання якості даних, очевидно, потрібні специфічні та чітко визначені показники. Також, ці показники повинні охоплювати різні аспекти якості [16].

Унікальність. Унікальність – вимоги до того, щоб об'єкти, змодельовані в межах підприємства, фіксувалися та представлялися однозначно. Ствердження унікальності сутностей у наборі даних означає, що жодна сутність не існує більше одного разу в наборі даних та існує ключ, який можна використовувати для однозначного доступу до кожної сутності (і лише до цієї конкретної сутності) у наборі даних [17]. Наприклад, у головній таблиці товару кожен продукт повинен з’являтися один раз і йому повинен бути присвоєний унікальний ідентифікатор, який представляє цей продукт у клієнтських програмах. Цей вимір можна контролювати двома способами. Як статична оцінка, це передбачає застосування дублюючого аналізу до набору даних, щоб визначити, чи існують дублікати записів, а як постійний процес моніторингу, він передбачає надання послуги збігу ідентифікації та роздільної здатності під час створення записів для пошуку точних або потенційних записів відповідності.

Точність. Точність даних – міра, з якою дані правильно відображають об’єкти “реального життя”, для моделювання якого вони призначані. У багатьох випадках точність вимірюється тим, як значення узгоджуються з визначеним джерелом правильної інформації (наприклад, довідковими даними). Існують різні джерела правильної інформації: база даних записів, подібний підтверджувальний набір значень даних з іншої таблиці, динамічно обчислювані значення або, можливо, результат ручного процесу.

Консистентність. У своїй базовій формі, консистентність відноситься до узгодженості значень даних в одному наборі даних, зі значеннями в іншому наборі даних. Суворе визначення консистентності вказує на те, що два значення

даних, отримані з окремих наборів даних, не повинні суперечити один одному, хоча консистентність не обов'язково передбачає правильність. Ще більш складним є поняття консистентності з набором заздалегідь визначених обмежень. Однак слід бути обережним, щоб не плутати консистентність з точністю чи правильністю. Консистентність може бути визначена в різних контекстах:

– між одним набором значень атрибутів та іншим набором атрибутів у тому самому записі (консистентність рівня запису);

– між одним набором значень атрибутів та іншим набором атрибутів у різних записах (консистентність перехресних зв’язків);

– між одним набором значень атрибутів та одним і тим же набором атрибутів в межах одного запису в різні моменти часу (темпоральна консистентність);

– консистентність може також враховувати концепцію "обґрунтованості", коли певний діапазон прийнятності застосовується до значень набору атрибутів.

Повнота. Очікування повноти вказує на те, що певні атрибути об’єкту повинні мати значення у кожному елементі набору даних. Правила повноти можуть бути призначені набору даних у трьох рівнях обмежень:

– обов’язкові атрибути, що мають бути задані;

– необов’язкові атрибути, які можуть мати значення на основі певного набору умов;

– непридатні атрибути, які можуть не мати значення.

Повнота також може розглядатися, як охоплення зручності та відповідності значень даних. Наприклад, щоб забезпечити доставку всіх замовлень, кожна позиція повинна посилатися на товар, а кожна позиція повинна мати ідентифікатор товару. Отже, позиція недійсна, якщо не заповнене поле “Ідентифікатор товару” [18].

Відповідність. Цей вимір стосується того, чи екземпляри даних зберігаються, обмінюються або подаються у форматі, який відповідає домену значень, а також сумісний з іншими подібними значеннями атрибутів [19]. З

кожним стовпцем пов’язані численні атрибути метаданих: тип даних, точність, шаблони форматування, використання заздалегідь визначеного переліку значень, діапазонів доменів, основних форматів зберігання тощо.

Цілісність посилань. Присвоєння унікальних ідентифікаторів об'єктам (клієнтам, продуктам тощо) у сховищі даних спрощує управління даними, але вносить нові очікування щодо того, що будь-коли, коли ідентифікатор об'єкта використовується як зовнішні ключі в наборі даних для посилання на основне представлення, що основне представлення насправді існує. Більш формально це відноситься до посилальної цілісності, і правила, пов'язані з посилальною цілісністю, часто виявляються як обмеження проти дублювання (щоб гарантувати, що кожна сутність представлена один раз і лише один раз), і правила цілісності посилань, які стверджують, що всі значення використовували всі первинні ключі фактично посилаються на існуючий основний запис.

Детальне дослідження даних сприяє розумінню їх семантики та структури [20]. Команда, що займається міграцією даних може оцінювати невідповідності, надмірності, неточності та цілісність посилань у таблицях та атрибутах, зменшуючи, наприклад, ризик пошкодження даних. Крім того, зменшується кількість переробок даних, що запобігає затримкам під час розробки або підтримки проекту. Як побічний продукт аналізу, характеристики проекту можуть бути використані для більш точного вирішення ризиків IT-управління внаслідок затримок проекту та перевитрати бюджету.

Отже, перевірка даних стосується таких аспектів: унікальність, точність, послідовність, повнота, відповідність, цілісність посилань, а також взаємодію з іншими додатками. Таким чином, для перевірки даних та їх структури має застосовуватися складна комбінація автоматизованих та ручних перевірок. Основною проблемою є порівняння даних із двох джерел з урахуванням заздалегідь визначених критеріїв порівняння.

Зазвичай, пакет тестів включає в себе комбінацію перевірок вказаних вище аспектів, тобто, при виконанні перевірки окремого елементу даних, перевіряється його унікальність (унікальність первинного ключа), повнота (чи має даний об’єкт

усі необхідні атрибути), послідовність (наприклад, статус замовлення “оплачено”, за умови наявності достатньої кількості транзакцій), відповідність (чи входять значення полів до області дозволених) та цілісність посилань (наприклад, усі товари, що входять до замовлення, повинні існувати).

Такі тести, які зазвичай називають тестами узгодженості, обробляють тисячі об'єктів у вихідному та цільовому сховищах даних [21]. Скрипти даних тестів автоматично конвертують первинні ключі (відповідно ідентифікатори) вихідного та цільового сховищ для забезпечення повноти.

Зіставлення даних між джерелами та цільовими ідентифікаторами є складним, якщо ідентифікатори або структура даних змінюються під час міграції даних. Тоді такі тести мають імітувати ці зміни. Імітаційний код не слід копіювати з програм фільтрації та трансформації, які перевіряються звіренням. Натомість слід розробити новий код, щоб не повторювати минулі концептуальні помилки або помилки на етапі впровадження. Ключовими результатами тестів на повноту та відповідність типу є, по-перше, список із усіма відсутними об’єктами в цільовій програмі та, по-друге, список із усіма об’єктами, які знаходяться в цільовій програмі, хоча їх не повинно існувати.

Для перевірки та оцінювання якості міграції потрібно вивести ряд критеріїв. Дані критерії можна розділити на швидкісні та якісні, бо у цілому успішність процесу міграції визначається якістю перенесених даних, а витрати на її проведення – швидкістю її виконання.

До критеріїв якості можна віднести:

– відносна точність – відношення загальної кількості рядків у вхідному сховищі до кількості рядків у цільовому, після міграції, може приймати значення від 0 до 1;

– відносне відхилення – відношення загальної кількості рядків у вхідному сховищі до кількості рядків, що мають викривленні дані у цільовому, після міграції, може приймати значення від 0 до 1.

– загальна кількість втрачених даних – представляю собою загальну кількість рядків, що присутя у вхідній СУБД, але відустня у цільовій;

– загальна коректність зв’язків – кількість рядків, що не проходять перевірку цілісності (тобто мають порушені зв’язки) у цільовій СУБД;

– загальна кількість рядків з відсутніми атрибутами – загальна кількість рядків, що мають один чи більше відсутніх обов’язкових атрибутів;

– загальна кількість рядків, що мають атрибути з пошкодженими значеннями (наприклад текстові значення з різною довжиною у вихідному та цільовому сховищах).

2.5.2 Проектування бази даних для дослідження

Для проведення експериментального дослідження є необхідність проектування та наповнення бази даних, що, у наслідку, буде використовуватись для тестування різних методів міграції даних.

Для прикладу, можна взяти предметну область обліку автомобільних запчастин, бо вона є достатньо складною та обширною.

На етапі проектування треба окреслити опорні сутності, виходячи з яких будуть будуватися зв’язки:

– виробник автомобіля, має посилання на ім’я та логотип, описується таблицею “Manufacturers”;

– модель автомобіля, має назву (наприклад: “A6”, що в поєднанні з назвою виробинка виглядає як “Audi A6”), дати початку та кінця виробництва, ім’я та зображення, описується таблицею “Models”;

– підмодель автомобіля, представляє собою окрему варіацію моделі (комплектацію, варіант двигуна, тощо.), має назву (наприклад: “1.4 TSI Touring ZF”, що в поєднанні з назвою виробинка виглядає як “Audi A6 1.4 TSI Touring ZF”), дати початку та кінця виробництва, ім’я та зображення, описується таблицею “Submodels”;

– виробник запчастини, має назву та логотип, описується таблицею “Brands”;

– поєднання артикульного номеру запчастини з її виробником, описуєтсья таблицею “ArticleIdentities”;

– коренева запчастина, що зазвичай невидима кінцевому користувачу, бо продається вона через “OEM” виробників (“виробників комплектного обладнання”), має поєднання артикульного номера з виробником, короткий опис та зображення, описується таблицею “Articles”;

– описання характеристики запчастин, скадається з назви та одиниці виміру, описується таблицею “ArticleCharacteristics”;

– запчастина, яку бачить кінцевий користувач, є поєднанням артикульного номеру, виробника «виробників комплектного обладнання», та типу (оригінальна чи післяринкова), описується таблицею “ArticleLookup”, може містити у собі множинні посилання на кореневі запчастини;

– категорія запчастин, містить назву, логотип та опціональне посилання на батьківську категорію (дозволяє створити ієрархію), описується таблицею “Categories”;

– автомобільний двигун, має назву, код, виробника та дати виробництва, описується таблицею “Engine”;

– набір базових характеристик, що є у кожного окремого двигуна (наприклад: очікувана максимальна потужність у кінських силах та кіловатах), описується таблицею “EngineBasicCharacteristic”;

– додаткова опціональна характеристика двигуна, має назву та одиницю виміру, описується таблицею “EngineCharacteristic”;

– набір базових характеристик підмоделі автомобіля (наприклад: кількість дверей, наявність ABS, тип трансмісії, тощо), описується таблицею “SubmodelBasicCharacteristic”;

– додаткова опціональна характеристика підмоделі, має назву та одиницу виміру, описується таблицею “SubmodelCharacteristic”;

– додатковий описовий дескриптор підмоделі, включає в себе VIN-код, код двигуна, заводські параметри шин та код кузова, описується таблицею “SubmodelSearchDescriptors”.

Також повинні бути окреслені допоміжні сутності, які не мають відношення до предментної області:

– текстовий ресурс, посилання на який міститься майже у всіх сутностях та зберігає текст, що буде відображатися за замовчуванням, без використання специфічної мови, описується таблицею “TextResource”;

– переклад текстового ресурса на ту чи іншу мову, описується таблицею “TextResourceTranslation”;

– мова, описується таблицею “Language”;

– графічний ресурс, посилання на який міститься у сутностях, що мають графічну інтерпретацію, зберігає у собі посилання на зображення, описується таблицею “GraphicalResource”.

Повинні бути промодельвані зв’язки між опорними сутностями за принципом “багато-до-багатьох”:

– один двигун може мати множину характеристик із значеннями та одна характкеристика може описувати множину двигунів, що описується таблицею “EngineCharacteristicValue”;

– одна підмодель може мати множину характеристик та одна харктеристика може описувати множину підмоделей, що описується таблицею “SubmodelCharacteristicValue”;

– одна коренева запчастина може бути вкладена у множину категорій та одна категорія может містити множину запчастин, що описується таблицею “ArticleCategoryLink”;

– одна коренева запчастина может мати множину характеристик із значеннями та одна характеристика може описувати множину запчастин, що описується таблицею “ArticleCharacteristicLinks”;

– одна запчастина може бути застосована до множини підмоделей та одна до одної підмоделі може бути застосована множина запчастин, що описується таблицею “ArticleSubmodelLinks”;

– одна підмодель може вироблятися із різними ревізіями двигунів під час її життєвого та один двигун може встановлюватися на зовсім різні підмодели, що описується таблицею “SubmodelEngineLink” (див. рис. 2.4).

2.5.3 Вибір технічних засобів для проведення експериментального дослідження

Для тестування міграції даних з використанням ETL та реплікації, потрібно обрати технічні засоби для їх реалізації.

Існує багато інструментів, які дозволяють реалізувати реплкацію даних між фундаментально різними СУБД, але зазвичай вони є комерційними і потребують ліцензії або підписки для використання. Одним з небагатьох безкоштовних рішень є програмний комплекс SymmetricDS, що має вбудовану підтримку багатьох популярних СУБД, включаючи ті, на які націлене дослідження – MySQL та Microsoft SQL Server.

SymmetricDS - це програмне забезпечення з відкритим вихідним кодом для синхронізації баз даних, що підтримує асинхронну реплікацію з декількома головними вузками, фільтровану синхронізацію та перетворення. Зміни можна тиражувати як заплановану операцію або операцію в режимі реального часу, також програмний комплекс включає функцію початкового завантаження для повної міграції даних. Також, SymmetricDS підтримує горизонтальне масштабування та роботу на вузлах з низькою пропускною можливістю. Засіб написаний на Java і вимагає стандартного випуску середовища виконання Java (JRE) (SE) або Java Development Kit (JDK) Standard Edition (SE) версії 8.0 або вище. Підтримуються більшість основних операційних систем та баз даних.

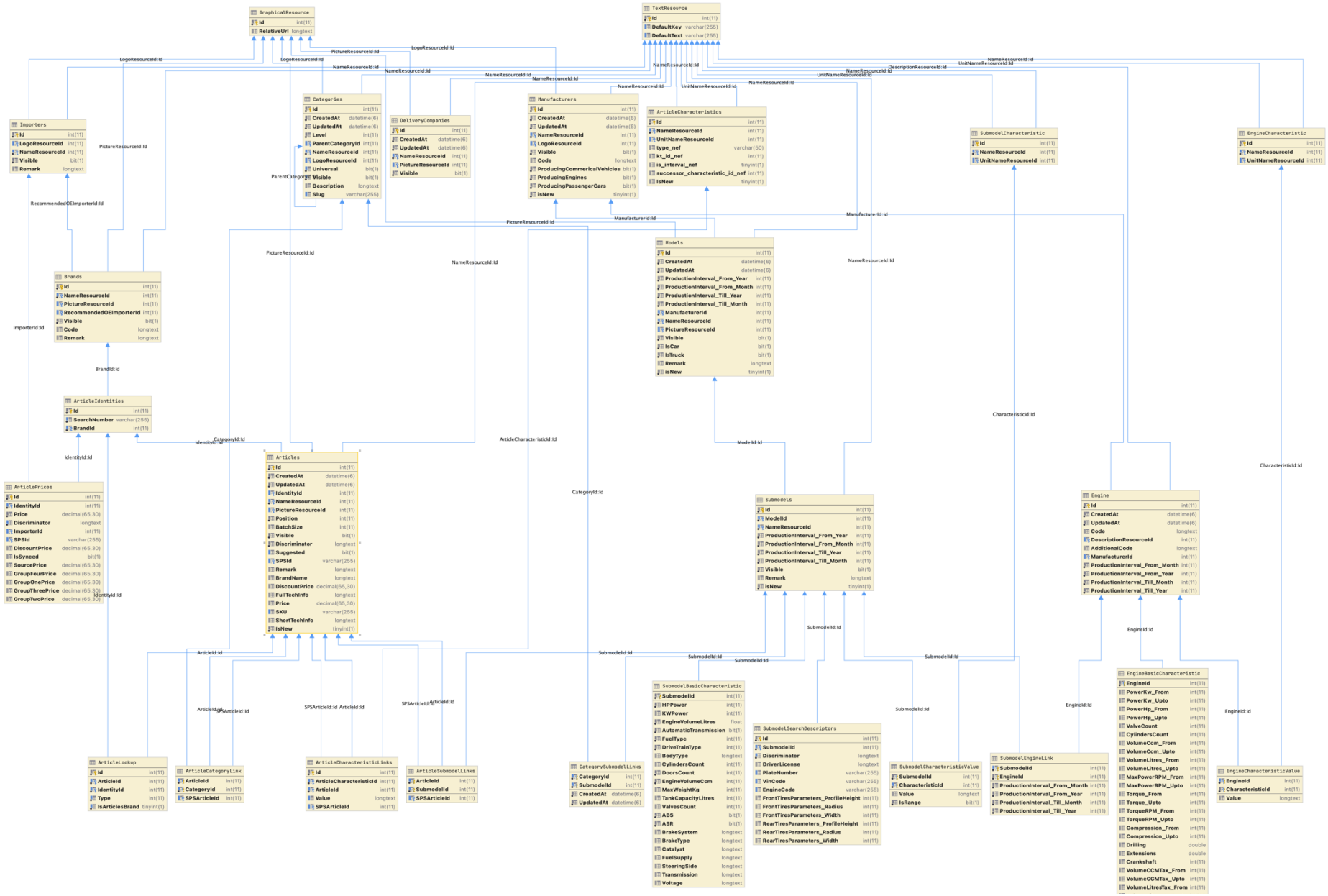

Рисунок 2.4 – Діаграма бази даних

Вимоги до пам'яті, диску та процесору зростають із збільшенням кількості підключених клієнтів та обсягу синхронізованих даних.

Вузол відповідає за синхронізацію даних з бази даних або файлової системи з іншими вузлами в мережі за допомогою HTTP. Вузли присвоюються одній із груп вузлів, які налаштовані разом як окрема самостійна одиниця. Групи вузлів пов'язані між собою з посиланнями на групи для визначення push або pull зв'язку. Pull призводить до того, що один вузол підключається до інших вузлів і запрошує зміни, що відбулися на інших вузлах, тоді як при pull – вузол підключається до інших вузлів, коли йому потрібно надіслати зміни.

Для тестування підходу ETL теж існує багато рішень, але, знову ж, всі вони є комерційними та не мають безкоштовних версій.

Програмна система Talend Open Studio є одним з небагатьох безкоштовних інструментів ETL для інтеграції даних, доступних на ринку. Вона дозволяє легко керувати всіма етапами, що беруть участь у процесі ETL, починаючи від створення проекту і закінчуючи завантаженням даних. Програмна система надає графічне середовище, за допомогою якого можна легко зіставити дані між джерелом і цільовим сховищем. Процес роботи складається з перетягування потрібних компонентів у область проекту, їх налаштуваня і з’єднання між собою. Виходячи з цього, можна зробити висновок, що даних програмний комплекс забезпечує швидку інтеграцію даних, а також надійний та швидких процес ETL.

3 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

3.1 Розробка програмного засобу для тестування якості міграції

Для отримання максимально правильних та достовірних результатів під час проведення дослідження потрібно мати надійне джерело інформації, що буде давати консистентний та надійний результат на одному наборі даних.

Для цього була розроблена програмна система, що порівнює дані у різних системах управління базами даних, що мають однакову схему (однакову кількість таблиць та колонок у таблицях).

Програмна система сканує службові таблиці у вихідній базі даних [22], формує віртуальну модель схеми у пам’яті та тестує цільову базу даних на «рівність» вихідній. Під словом «рівність» мається на увазі сукупність фактів, що вказує на те, що зміст цільової бази даних є дзеркальною копією вихідної бази даних.

Додаток реалізованій за допомогою мови C#, TypeScript, фреймворків ASP.NET та Angular з застосуванням технології WebSockets. Клієнтська частина розроблена за допомогою Angular із використанням бібліотеки Socket IO для реалізації постійного зв’язку та синхронізації результатів із серверною частиною. В свою чергу, серверна частина розроблена за допомогою ASP.NET з використанням бібліотек Dapper, що дозволяє власне виконувати команди до СУБД, та SignalR, що дозволяє реалізувати WebSockets для підтримки зв’язку з клієнтом у режимі реального часу.

Додаток працює за принципом створення фонових робіт та їх конвеєрної обробки [23], кожна із перевірок виконана у вигляді окремого етапу.

Власне процес перевірки полягає у збірці результатів порівняння баз даних у двух СУБД за критеріями вказаними у розділі 2.5.1. Одним з етапів, не вказаних у роздлі 2.5.1, є порівняння схем вихідної да цільової баз даних, що виконується за допомогою збірки метаданих таблиць у кожній з СУБД за допомогою системних схем, що містять інформацію про структуру баз даних.

Для використання додатку достатньо натиснути на «Створити роботу» (див рис. 3.1), ввести дані цільових СУБД, обрати необхідні перевірки та нативснути «Старт» (див рис. 3.2). З цього моменту починається власне перевірка баз даних, її прогрес можна побачити у вікні «Прогрес» (див рис. 3.3) обравши потрібну роботу за назвою та датою почкатку. Після повного виконання перевірки можно отримати зведення її результатів у розділі «Звіти», там же можна сгенерувати «pdf» файл з результатами перевірки

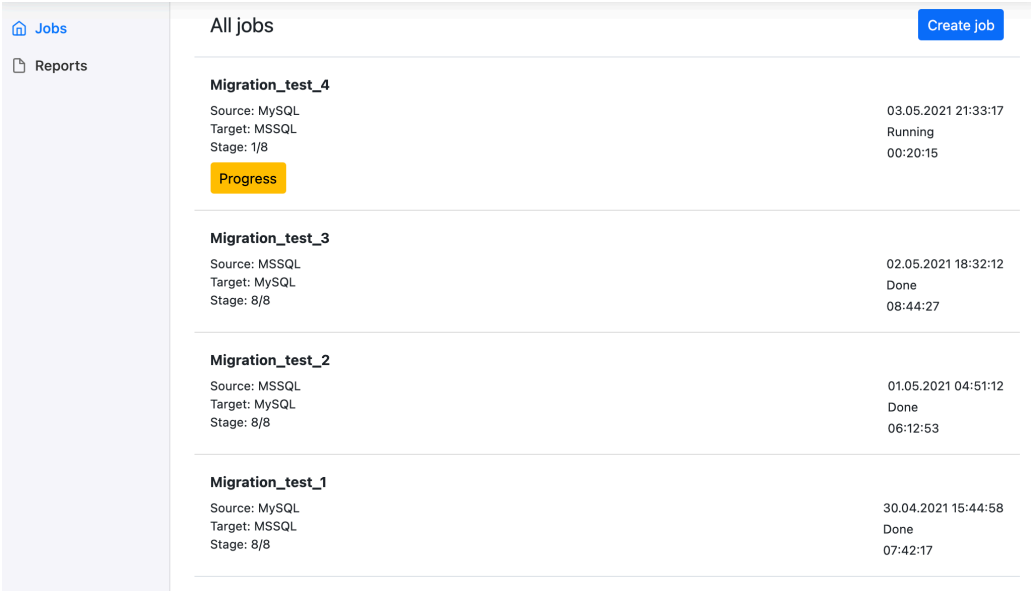

Рисунок 3.1 – Основний інтерфейс програми

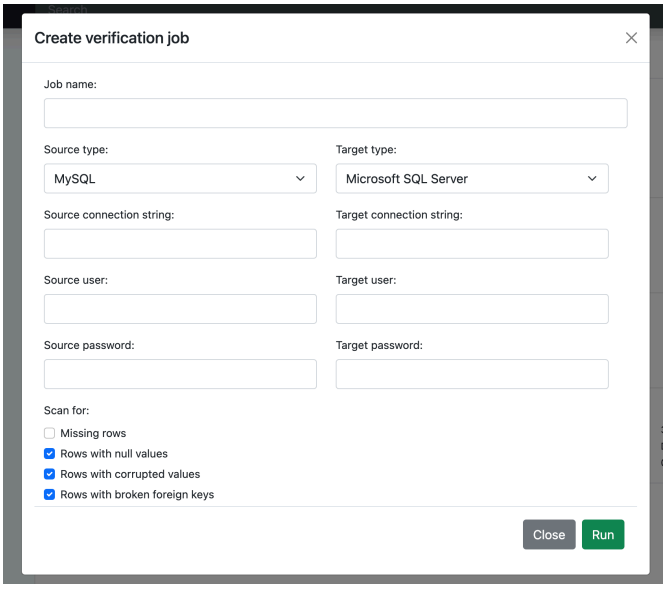

Рисунок 3.2 – Створення роботи

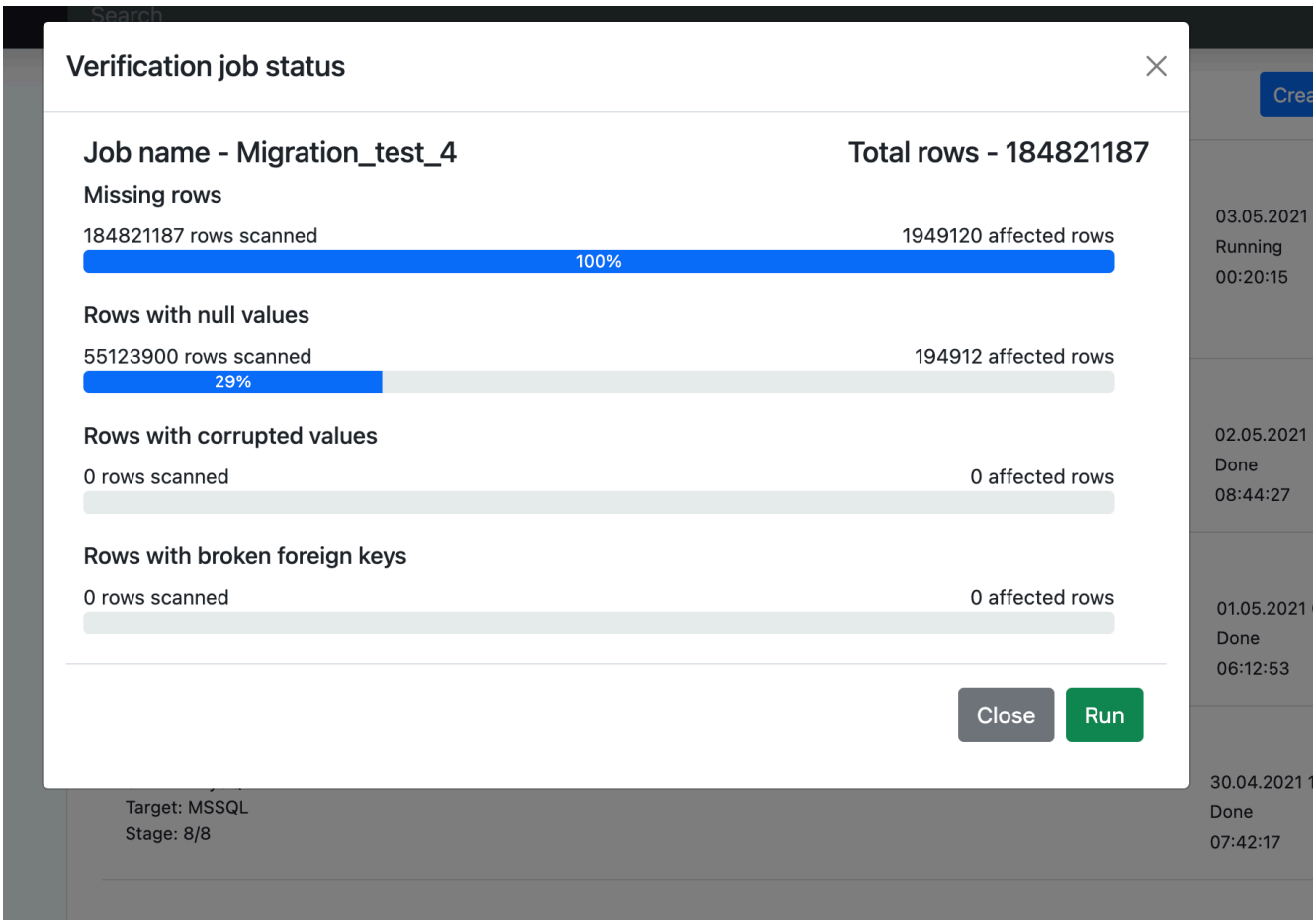

Рисунок 3.3 - Вікно «Прогрес»

На рівні реалізації програмна система може представлена за допомогою декількох логічно розділених підсистем. Складовими основними підсистеми є:

– VerificationController – контроллер, який є основною точкою взаємодії між клієнтом та сервером за допомогою звичайних HTTP запитів;

– VerificationHub – SignalR хаб, який дозволяє клієнту отпримувати від сервера дані про процес перевірки у реальному часі;

– BackgroundJobManager – сервіс, що відповідає за послідовність виконання запущених робіт та можливість їх запуску паралельно;

– BackgroundJob – атомарна одиниця роботи, в даному випадку екземпляр запущеної або закінченої перевірки;

– VerificationPipelineExecutor – відповідає за правильність очередності запуску етапів перевірки;

– JobProcessObserver – дозволяє клієнту у реальному часі отримувати дані про процес перевірки на кожному з етапів;

– ReportGenerationStrategy – абстрактна стратегія генерації звітів з проведених перевірок.

Власне імплементація різних етапів перевірки не відрізняється з точки зору організації класів, але відрізняється з точки зору їх реалізації, основними її складовими є:

– VerificationContext – контекст, що є спільним для усіх етапів перевірки, містить статус та результати кожного з етапів.

– VerificationStage – абстрактний обробник етапу перевірки, є точкою його запуску та містить усю логіку;

– CallbackExecutor – збирає дані, про нинішній статус перевірки та надає можливість передати його клієнту.

Додаток побудований на основі клієнт-серверної архітектури (див. рис 3.4), користувач взаємодіє з так званим «тонким клієнтом», що запущений у браузері. Серверна частина виконує усі операції та зв’язується із двома базами даних, із цільовою та вихідною.

3.2 Розробка самописаного програмного засобу для проведення міграцій

Розробка програмного засобу для проведення міграції даних є складним завданням, бо воно включає в себе велику кількість труднощів, що є складними у вирішенні. При розробці самописаного рішення для міграції були вирішені такі проблеми:

– перенесення даних у коректному порядку для підтримання посилкової цілосності;

– корректне перенесення текстових даних;

– корректне пересення даних у байтових типах (BLOB);

– корректне перенесення числових даних (з урахуванням кількості знаків після коми).

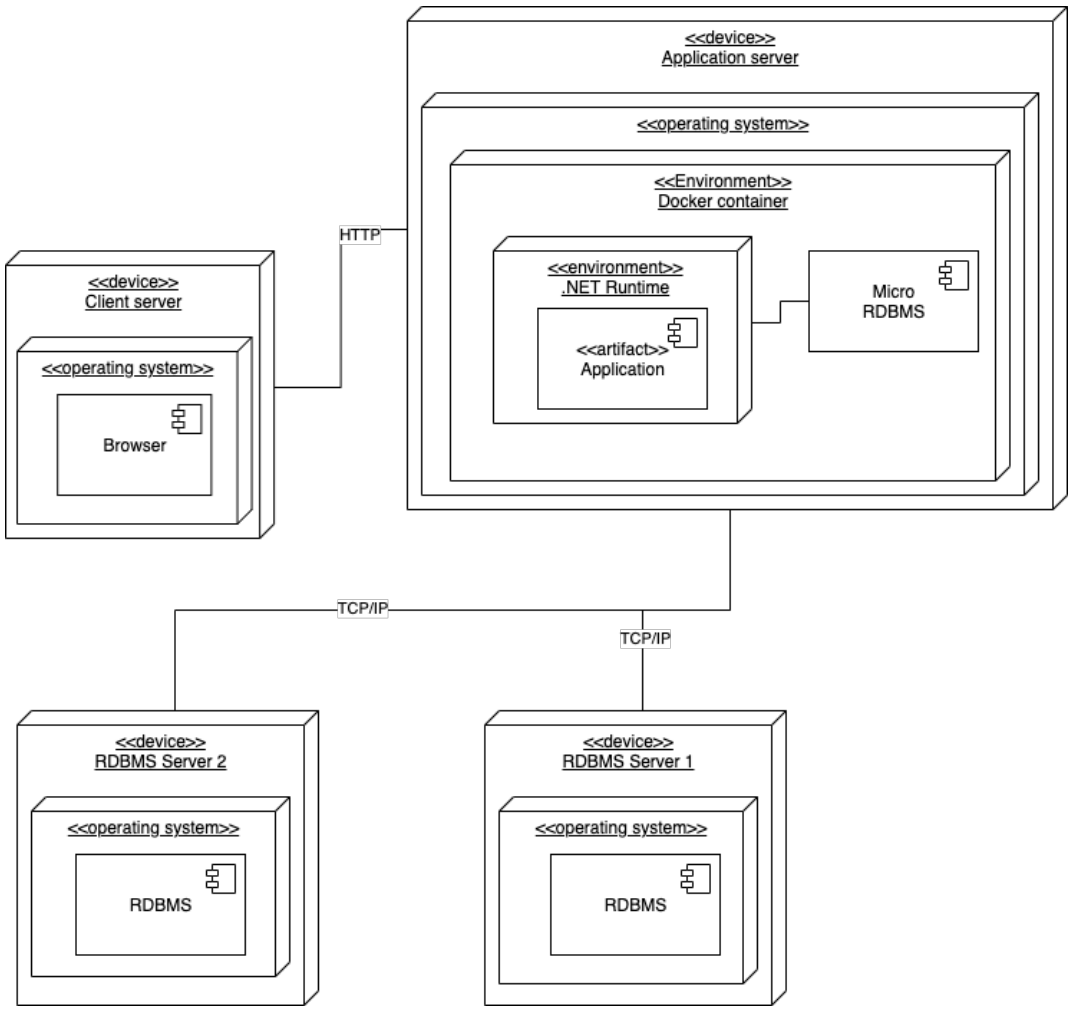

Рисунок 3.4 – Діаграма розгортування додатка

Програмний засіб виконаний у вигляді утиліти командного рядка (див. рис. 3.5), що приймає декілька параметрів, обов’язкові - рядок з'єднання для вихідної СУБД та рядок з’єднання для цільової СУБД, необов’язкові – кількість рядків, що переноситься за один прохід (“bulk-row-count”), максимальна кількість потоків, що може одночасно виконувати операції виборки/вставки (“max-parallel-threads”).

```
➜  exec ./dbMigrator --bulk-row-count 15000 --max-parallel-threads 8 \
--source "Server=localhost;Database=testDatabase;User Id=user;Password=pwd;" \
--target "Server=localhost;Database=testDatabase;Uid=user;Pwd=pwd;" \
--type "mssql-to-mysql";
```

Рисунок 3.5 - Приклад консольної команди для запуску міграції

Додаток реалізованій за допомогою мови C# та бібліотек-драйверів для взаємодії з MySQL та MSSQL за домогою ADO.NET.

Додаток може бути запущеній у будь-якій середі де є підтримка середи виконання .NET 5, щодо підтримки СУБД – додаток підримиує MySQL починаючи з версії 8 та Microsoft SQL Server починаючи з версії 2017.

Після запуску, програмний засіб перевіряє наявність підключеннях до обох СУБД та починає процес міграції [24]. Власне процес міграції складається з декількох етапів:

- отримання метаданих;
- перевірка схем на «рівність»;
- формування черговості перенесення даних за таблицями;
- власне перенесення даних;
- формування звіту з міграції.

Отримання метаданих про бази даних - повний прохід по структурі таблиць та зв’язків між ними.

Перевірка схем на «рівність» - перевірка ріності кількості колонок у таблицях, їх типів даних (беручи до уваги сумісність між MySQL та MSSQL),.

Формування черговості перенесення даних за таблицями – невеликий етап, що формує чергу, за якою будуть переноситися дані, першими йдуть таблиці без зовнішніх зв’язків, потім – інші [25].

Перенесення данних, у свою чергу, складається з двух етапів – витяг із вихідної СУБД за допомогою оператора SELECT та вставка у цільову БД за допомогою оператора INSERT. Ці операції виконуються пакетно, тобто, за одне виконання SELECT/INSERT переноситься кількість рядків, що вказана у параметрі “bulk-row-count” (значення за замовчуванням - 1000), також, для таблиць, що не мають зовнішніх ключів, дані операції виконуються паралельно (максимальна кількість потоків регулюється параметром “max-parallel-threads”).

Після виконання міграції програмна система формує звіт у вигляді тексту, що містить дані, щодо переносу даних кожної таблиці. Формат даних показаний на рисунку 3.6.

```
[110][R][S] S: 'ArticleCategoriesLink' SIZE: 15000 OFFSET 165000
[125][W][S] S: 'ArticleCategoriesLink' T: 'dbo.ArticleCategoriesLink' SIZE: 15000
[135][R][S] S: 'ArticleCategoriesLink' SIZE: 15000 OFFSET 180000
[127][W][S] S: 'ArticleCategoriesLink' T: 'dbo.ArticleCategoriesLink' SIZE: 15000
[129][R][S] S: 'ArticleCategoriesLink' SIZE: 15000 OFFSET 195000
[145][W][F] S: 'ArticleCategoriesLink' T: 'dbo.ArticleCategoriesLink' SIZE: 15000 ERROR: 'Cannot resolve the collation conflict between...'
```

Рисунок 3.6 – Процес виконання міграції

Найважливішими складовими даної системи з точки зору реалізації є такі класи:

– DatabaseTraverser – відповідає за первинне сканування метаданих баз даних;

– TableChecker – відповідає за оцінку сумісності таблиць різних СУБД;

– ColumnCompatibilityChecker – відповідає за оцінку сумісності типів колонок у окремій таблиці;

– JobQueue – відповідає за управління чергою задач, що виконуються у процесі міграції;

– ParallelCommandExecutor – відповідає за паралельне виконання декількох задач, якщо це можливо;

– MigrateTableCommand, MigrateTableCommandHandler – власне сама команда, що здійснює міграцію окремої таблиці у рамках міграції;

– InsertCommand, InsertCommandHandler – команда та обробник, що реалізуються операцію INSERT у тій чи іншій СУБД;

– SelectCommand, SelectCommandHandler – команда та обробник, що реалізуються операцію SELECT у тій чи іншій СУБД.

Додаток є повністю переносним та самостійним, не потребує встановлення і може бути запущений на будь-якій операційній системі завдяки однофайловій компіляції середи .NET, але слід зазначити, що для кожної операційної системи повинен бути скомпільований свій виконуваний файл.

4 ОПИС ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

4.1 Опис процесу першочергового завантаження даних у тестовані СУБД

Дані були згенеровані згідно із схемою вказаною на рисунку 2.4 випадковим чином у об’ємі 7 гігабайт та кількістю рядків в 184 мільйони (див. табл. 4.1) з використанням різноманітних типів даних у форматі CSV.

Таблиця 4.1 – Кількість тестових даних сгенерованих для кожної таблиці

Назва таблиці	Кількість рядків	Розмір CSV
ArticleLookup	72276616	2,82 ГБ
ArticleCategoryLink	27592128	520 МБ
ArticleSubmodelLinks	25794337	478,2 МБ
ArticleCharacteristicLinks	21060768	857,8 МБ
ArticleIdentities	18171428	522,7 МБ
TextResource	6665103	288,3 МБ
Articles	6635621	933,4 МБ
GraphicalResource	3442937	178,9 МБ
SubmodelSearchDescriptors	2880950	326,6 МБ
EngineCharacteristicValue	173647	5,1 МБ
TextResourceTranslation	25277	1,2 МБ
Engine	23068	2,1 МБ
EngineBasicCharacteristic	23068	1,9 МБ
SubmodelCharacteristicValue	20697	671 КБ
Submodels	7585	527 КБ
SubmodelBasicCharacteristic	7497	795 КБ
SubmodelEngineLink	4946	192 КБ
Models	4338	689 КБ
Categories	3809	412 КБ

Кінець таблиці 4.1

Назва таблиці	Кількість рядків	Розмір CSV
ArticleCharacteristics	3677	138 КБ
Brands	3375	126 КБ
Manufacturers	284	35 КБ
SubmodelCharacteristic	15	1 КБ
EngineCharacteristic	11	2 КБ
Language	5	1 КБ
Усього	184821187	6,94 ГБ

Для MySQL та Microsoft SQL Server існують вбудовані оператори, що дозволяють завантажувати дані із CSV файлів.

Для MySQL – це LOAD DATA INFILE

```
LOAD DATA INFILE '\Volumes\Data\CSV\Articles.csv'
INTO TABLE Articles
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 ROWS;
```

А для Microsoft SQL Server – це BULK INSERT

```
BULK INSERT dbo.Articles
FROM '\Volumes\Data\CSV\Articles.csv'
WITH (fieldterminator = ',', rowterminator = '\n');
```

Дані були завантажені у таблиці у черзі починаючи з таблиць без зовнішніх зв’язків та закінчуючи таблицями із великою кількістю зв’язків.

Отже, маючи ідентичні вихідні бази даних можна переходити до тестування власне методів міграції даних.

4.2 Опис методики тестування підходів до міграції даних

При проведенні дослідження програмні засоби, СУБД MySQL та MSSQL були запущені на окремомих серверах у рамках однієї локальної мережі, щоб уникнути колізій при розділенні машинних ресурсів.

4.2.1 Опис методики тестування ETL

Для тестування методики ETL був використаний продукт Talend Open Studio та плагіни для MySQL та MSSQL. Даний метод гарантує гнучкість у виборі джерел даних для витягування чи вставки та трансформації даних у процесі переносу.

Був створений проект та були створені підключення до цільової та вихідої СУБД (див. рис. 4.1) за допомогою відкритих JDBC плагінів MySQL JDBC Connector та FreeTDS. Під час створення підключень програмний комплекс автоматично відсканував бази даних та запропонував зберегти структури таблиць для подальшого їх використання.

Рисунок 4.1 – Проект створений у Talend Open Studio

У рамках створеного проекту були створені дві роботи (Job), призначені для прямого копіювання даних між таблицями (див. рис. 4.2). При додаванні кожної таблиці, програмний комплекс автоматично створив мапи зв’язків між колонками (див. рис. 4.3), що майже у всіх випадках були правильними.

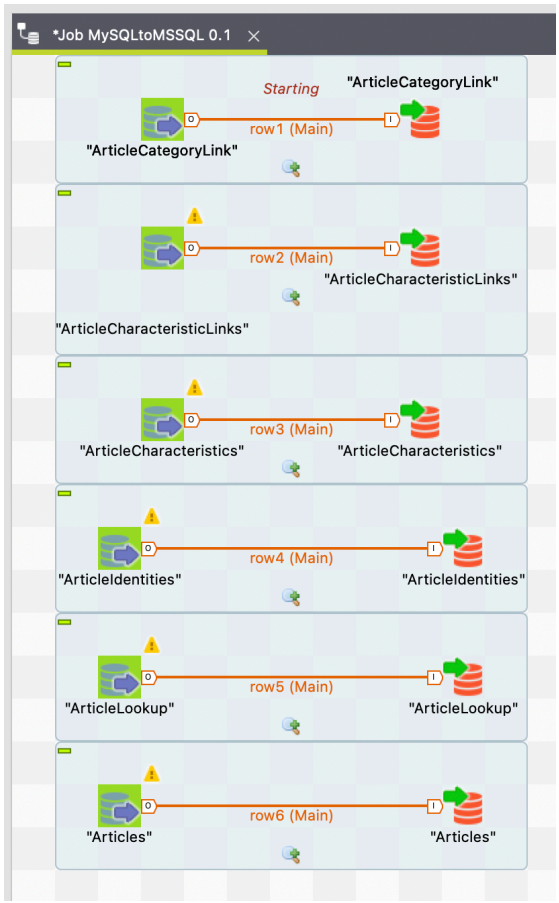

Рисунок 4.2 – Приклад роботи, створеної для міграції даних

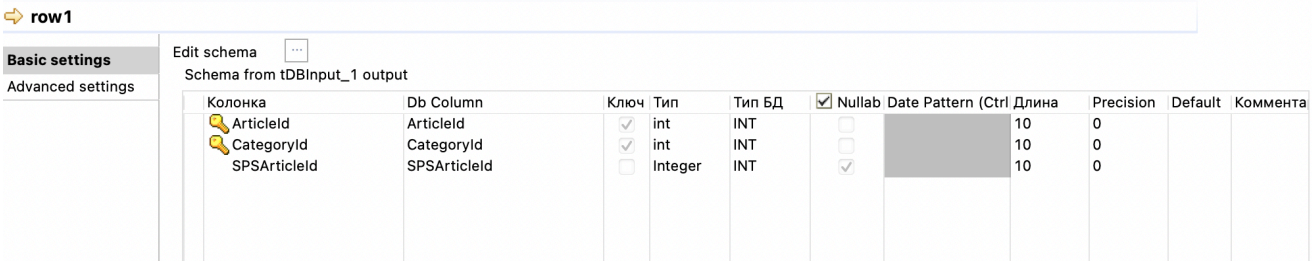

Рисунок 4.3 – Приклад мапи зв’язків колонок у таблицях різних СУБД

Також, слід зазначити, що під час перенесення даних у цільовій СУБД вимикається перевірка цілосності зовнішніх ключів, бо якщо цього не робити, то час імпорту займав би в декілька разів більше часу.

Після завершення процесу імпорту програмна система показує статистику по поточній роботі (див. рис. 4.4).

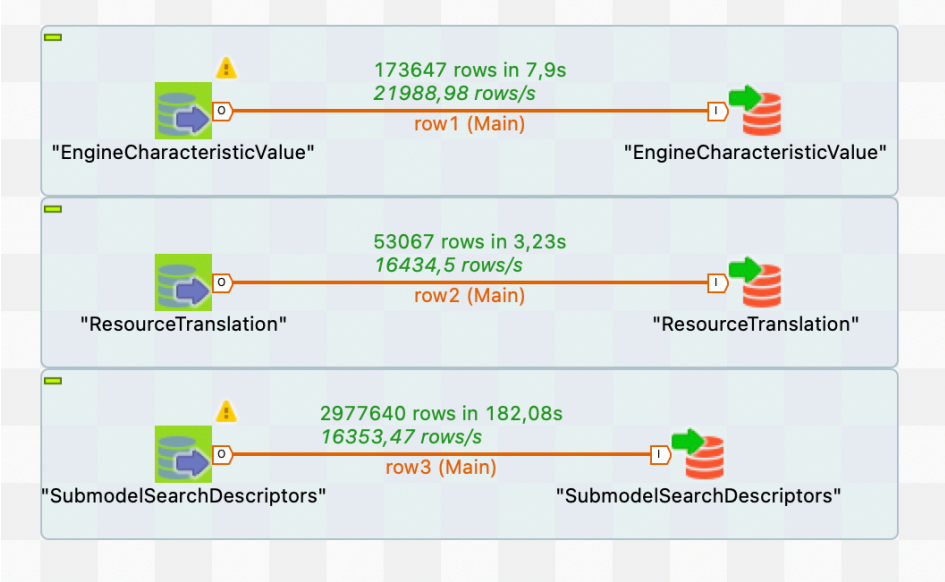

Рисунок 4.4 – Статистика по експортованим таблицям

4.2.2 Опис методики тестування реплікації

Для тестування міграції методом реплікації був обраних програмний комплекс SymmetricDS. Налаштування даного програмного косплексу відбувається за допомогою конфігураційних файлів з розширенням “properties”.

Кожен вузол підключений до бази даних за допомогою драйвера Java Database Connectivity (JDBC) за допомогою URL-адреси підключення, імені користувача та пароля. Під час запуску SymmetricDS шукає файли конфігурації вузлів (див. рис. 4.5) і запускає вузол для кожного знайденого файлу, що дозволяє декільком вузлам працювати на одному сервері та спільно використовувати ресурси. Файл властивостей вузла містить його зовнішній ідентифікатор, групу

вузлів, URL-адресу сервера реєстрації та інформацію про підключення до бази даних. Зовнішній ідентифікатор - це назва вузла, що використовується для його ідентифікації від інших вузлів. Коли вузол запускається вперше, він зв'язується із сервером реєстрації, використовуючи процес реєстрації, який надсилає його зовнішній ідентифікатор та групу вузлів. У відповідь вузол отримує свою конфігурацію та пароль вузла, який необхідно надіслати як аутентифікацію під час синхронізації з іншими вузлами.

```
engine.name=source-000
db.driver=net.sourceforge.jtds.jdbc.Driver
db.url=jdbc:sqlserver://192.168.0.201:1433;databaseName=test;integratedSecurity=true;
db.user=test
db.password=fnqkDPUHlGBZQ0i21i7M
registration.url=
sync.url=http://localhost:31415/sync/source-000
group.id=source
external.id=000
job.purge.period.time.ms=7200000
job.routing.period.time.ms=5000
job.push.period.time.ms=10000
job.pull.period.time.ms=10000
initial.load.create.first=false
```

```
engine.name=target-001
db.driver=com.mysql.jdbc.Driver
db.url=jdbc:mysql://192.168.0.202:3306/test
db.user=test
db.password=xuvs0Hgg4Ck92edB5gQ4
registration.url=http://localhost:31415/sync/source-000
group.id=target
external.id=001
job.routing.period.time.ms=5000
job.push.period.time.ms=10000
job.pull.period.time.ms=10000
```

Рисунок 4.5 – Приклади використаних конфігурацій SymmetricDS

Під час двух експериментальних проходів, у якості мастер-вузла була налаштована вихідна СУБД, а у якості підписника, тобто дочірнього вузка – цільова, а процес реплікації у цілому був запущена в односторонньому асинхронному режимі. Також, слід зазначити, що усі екземпляри вузлів запускалися на одному сервері, окремому від серверів, на яких були запущені СУБД. Після первісного налаштування та запуску мастер-вузлів, були запущені дочірні вузли, які власне і почали процес міграції даних.

Під час процесу виконання, SymmetricDS формує лог-файли, за допомогою яких можна відстежити наявність проблем у процесі міграції та її загальний час виконання.

4.2.3 Опис методики тестування самописного методу

Дослідження ефективності самописаного методу було здійснене з допомогою програмної системи розробленої та описаної в розділі 2.7. Максимальна кількість потоків була установлена в кількості 8, а максимальний розмір атомарного пакету даних – 15000.

4.3 Результати проведення експерименту

Кожен з підходів до міграції був протестований у двух напрямках з MySQL до Microsoft SQL Server та з Microsoft SQL Server до MySQL. Тести були запущені окремо та після кожного з них бази даних повністю видалялися, тому результати можно вважати достовірними.

Час виконання повної міграції даних у хвилинах (див. рис. 4.6) чітко дає зрозуміти, що програмні продукти сторонніх розробників є більш оптимізованими та швидшими, ніж рішення власної розробки.

З огляду на кількість втрачених рядків (див. рис. 4.7), можна зробити висновок, що при застосуванні самописного рішення була втрачена мінімальна кількість даних, але, основним чинником цього є те, що при відходженні від ідеальних умов власна розробке буде потребувати доробок та змін, бо вона не є универсальною та гнучкою. Якщо відштовхуватися від загальної кількості даних, то втрати не виглядають критичники (див. табл. 4.2).

Рисунок 4.6 – Діаграма часу виконання міграцій у хвилинах

Таблиця 4.2 – Кількість втрачених рядків у відсотках

MySQL в MSSQL			MSSQL в MySQL		
ETL	Реплікація	Самописне рішення	ETL	Реплікація	Самописне рішення
1,5%	0.001%	0,0007%	2%	0.002%	0,0011%

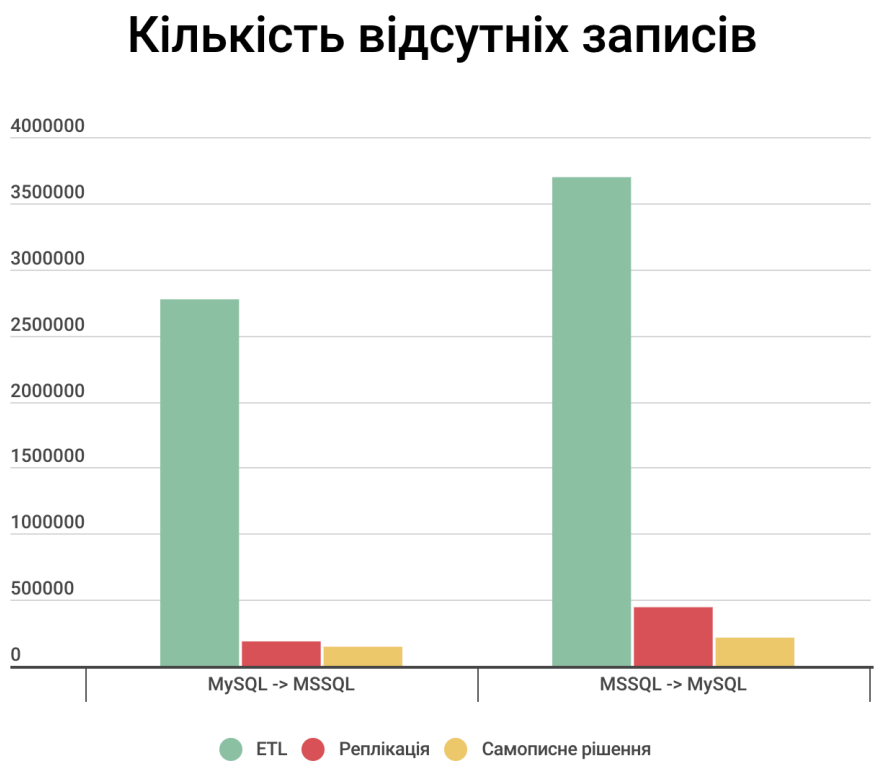

Рисунок 4.7 – Діаграма кількості втрачених рядків після міграції

Наступними критеріями оцінки є якість успішно перенесених даних, тобто цілісність зв’язків (див. рис. 4.8).

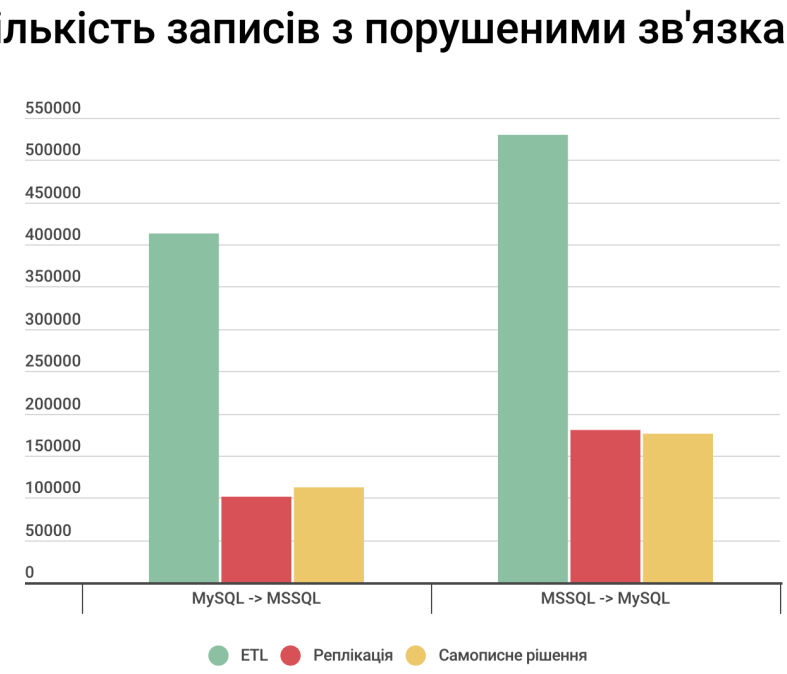

Рисунок 4.8 – Діаграма кількість рядків, що не проходять перевірку цілісності після міграції

Також були розглянуті критерії цілісності атрибутів у перенесених рядках – кількість рядків з відсутніми атрибутами (див. рис. 4.9), кількість рядків з пошкодженими атрибутами (див. рис. 4.10).

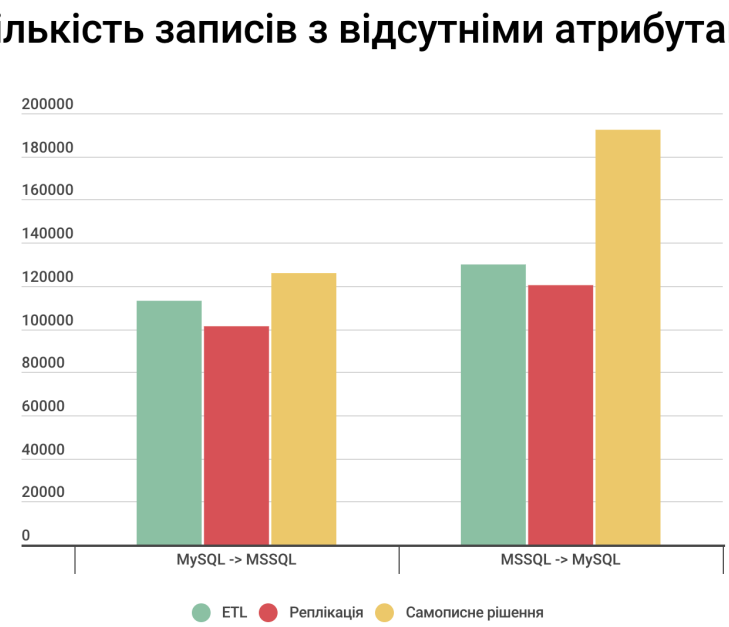

Рисунок 4.9 – Діаграма кількості рядків з відсутніми (NULL або пустими) атрибутами

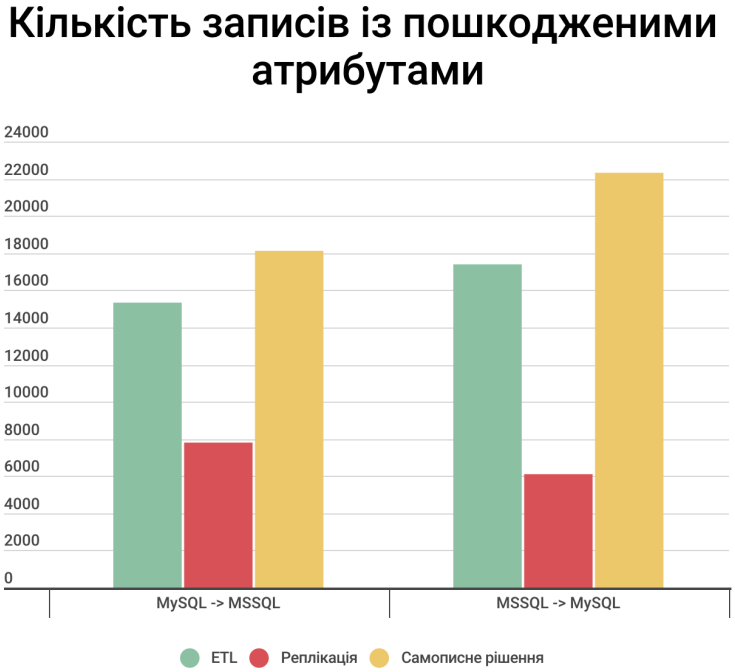

Рисунок 4.10 – Діаграма кількості рядків, що мають атрибути з пошкодженими значеннями

Велика кількість рядків з порушеними зв’язками обумовлена великою кількістю втрачених даних. Виходячи з даних результатів, можно зробити висновок, що підхід ETL не можна вважати надійним методом міграції даних, бо виправлення та корекція такої кількості даних потребує значної уваги та дуже великої кількості часу, також, через таку велику кількість «зломаних» записів, користувачам буде дуже важко (а інколи і неможливо) виконувати рутинні процедури у підключених до мігрованої бази даних програмних системах.

Втрати у якості самих рядків, а саме пошкодження та втрати деяких значень обумовлені деякими специфічними несумісностями між типами колонок у MySQL та Microsoft SQL Server. Наприклад, дана ситуація може виникнути при випадковій появі різниці у кодування текстових колонок, різниці у довжині числових типів, чи конвертації BLOB типів між різними СУБД. Також, виходячи з даних результатів, можна зробити висновок, що програмні засоби реалізації реплікації за ETL більш педантично та тонко обробляють дані ситуації, що у результаті дає меншу кількість рядків з пошкодженими даними.

4.4 Висновки та рекомендації з експериментального дослідження

З отриманих результатів можна отримати декілька важливих висновків.

Вибір методу міграції є дуже ситуативним та залежить від дуже багатьох факторів, основними з них є об’єм даних, режим роботи програмних систем, що використовують ці дані, бюджет та часові рамки.

Метод ETL показав найгірші результати, через його непристосованість до ситуацій прямих міграцій даних, тобто, фактично, етап трансформації фактично був пропущений, а порядок таблиць у процесі міграції повинен бути встановлений вручну, що додатково впливає на час планування та створює додаткову вірогідність появи помилок. Одною з позитивних сторін використання ETL для прямої міграції є наявність великої кількості існуючих інструментів, що,

в свою чергу, може дуже значно зменшити тривалість етапу розробки міграції, але, з іншого боку – більшість з них мають дуже високу вартість ліцензії. Даний метод більше підходить для сбору даних із різних джерел та їх конвеєрної обробки.

Метод реплікації показав себе, як достатньо надійну та швиду альтернативу до ETL та впровадження власних розробок. Серед основних переваг можно відзначити швидкість виконання міграції, можливість підтримки постійної синхронізації між різними СУБД та досить низьку кількість помилок.

Розробка власних інструментів для міграцій, у свою чергу, є достатньо доброю заміною для інших методів, найголовнішою перевагою якого є гнучкість. Експеримент показав, що, зазвичай, інструмент міграції власної розробки створює менше помилок (особливо втрачених рядків), але при цьому демонструє найменшу швидкість виконання. Не усі процеси міграції однакові та, за появи специфічних вимог до процесу міграції, існуючі інструменти можуть не підтримувати одну з критичних функцій, з іншого боку – деякі функції, що вже реалізовані у вже існуючих продуктах можуть бути неймовірно складними у реалізації. Також, додатковою слабкістю розробки власних програмних комплексів для міграції є висока вартість з точки зору часу, що включає в себе витрати на проектування та власне розробку.Слід звернути увагу, що втрати даних на невеликих об’ємах дуже легко відстежити завядки механізмам логування, саме тому для міграції невеликих об’ємів даних чудово підійде ETL, через його швидкість налаштування, але велику вірогідність появи помилок.

Можна використовувати комбінації із різних методів, для пришвидшення міграції та зменшення кількості дефектів даних, що виникають.

У вападку міграції великої кількості даних, добрим рішенням буде поєднання методів реплікації та розробки власного програмного комплексу. Таблиці, що мають критичне значення для бізнесу, можуть бути мігровані за допомогою інструментів власної розробки, що дасть повний контроль над процесом та гнучкість у змінах, а для інших таблиць підійде реплікація через свою швидкість та можливість синхронізації.

ВИСНОВКИ

В роботі розглянуті питання, пов'язані з міграціями даних, що зберігаються в реляційних базах даних, і процесом їх виконання. В рамках даної роботи досягнуті наступні результати:

– проаналізовані існуючі методи міграції даних;

– розглянута популярність методів міграції даних серед розробників;

– досліджений процес розробки плану міграцій;

– проаналізовані СУБД, між якими найчастіше проводяться міграції;

– розроблені критерії, що дозволять оцінити міграцію;

– розроблена схема бази даних для проведення експериментального дослідження;

– розроблені програмні системи для проведення міграції та оцінки їх якості;

– спланований та проведений експеримент, що дозволить дослідити ефективність різних типів міграцій між найпопулярнішими СУБД;

– сформовані рекомендації, щодо застосування методів міграції.

Вналсідок 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. Однак ними можна керувати та пом'якшувати за допомогою чітко визначеної стратегії та плану перевірки.

План міграції повинен включати оцінку даних, дослідження предметної області та визначення залежностей між об’єктами. Також слід звернути особливу увагу на залежності програмних засобів від даних, які зачіпає міграція, це також може призвести до проблем. Складний продукт може містити дуже багато різних об’єктів даних, що надходять із сотні різних додатків.

Є досить багато методів та підходів до міграції даних, але найпоширенішими з них є розробка та застосування власних комплексів, реплікація та ETL. Вони є фундаментально різними, але кожен з них має свої

переваги, хоча вони виконують практично одну й ту саму задачу тільки різними способами.

Вибір методу, у свою чергу повинен бути максимально можливо обгрунтованим, бо змінити рішення знаходячись на етапі впровадження буде дуже важко та майже неможливо. Зробивши невірний вибір чи впровадивши невірну комбінацію методів, можна поставити бізнес у скрутне становище, спотворивши чи знищивши історичні дані, що необхідні для аналітики.

Вибраний метод міграції, повинен поєднувати надійність, ефективність та здійснювати мінімальний вплив на кінцевих користувачів та бізнес-процеси.

Для подальших дослідженнях в сфері міграцій даних можна звернутися до методів міграції через хмарні середовища або методів роботи з Big Data.

За результатами роботу опубліковано тези доповіді «Дослідження проблем оцінки якості даних у міграціях даних» на конференції «Сучасні напрямки розвитку автоматизації, транспортних систем, технічних та комп’ютерних наук».

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: Architecture-Driven Modernization Case Studies – Morgan Kaufmann Publishers 2010. – 456 p.

6. Mark A. Dalton C. Multi-Domain Master Data Management: Advanced MDM and Data Governance in Practice – Morgan Kaufmann Publishers 2015. – 244 p.

7. David L. The Practitioner's Guide to Data Quality Improvement – Morgan Kaufmann Publishers 2010. – 432 p.

8. Michael G. Database Management Systems: Understanding and Applying Database Technology – Butterworth-Heinemann 1991. – 470 p.

9. Dave M.. Data Gravity – in the Clouds. Дата оновлення: 07.12.2010. URL: https://datagravitas.com/2010/12/07/data-gravity-in-the-clouds/ (дата звернення: 10.02.2021)

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

11. What is data migration? URL: https://www.netapp.com/knowledge-center/what-is-data-migration/ (дата звернення: 16.02.2021)

12. Tehreem Naeem. What is data migration. The Why, The What, and The How. Дата оновлення: 20.02.2021. URL: https://www.astera.com/type/blog/data-migration-software/ (дата звернення: 21.02.2021)

13. 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.

14. Stack overflow developer survey 2020. URL: https://insights.stackoverflow.com/survey/2020#technology-databases-professional-developers4 (дата звернення: 18.03.2021)

15. DB-Engines relational DBMS rankings. URL: https://db-engines.com/en/ranking/relational+dbms (дата звернення: 18.03.2021)

16. G. May, “Datenmigration,” Mu¨nchen, Germany, p. 35, 2010.

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

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

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

20. B. Wu, D. Lawless, J. Bisbal, R. Richardson, J. Grimson, V. Wade, and D. OSullivan, “The Butterfly Methodology : A Gateway-free Approach for Migrating Legacy Information Systems,” in 3rd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS97). Como, Italy: Institute of Electrical and Electronics Engineers, 1997, P. 200–205.

21. 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

22. I. Afanasieva, N. Golian, O. Hnatenko, Y. Daniiel, K. Onyshchenko, Data exchange model in the internet of things concept // Telecommunications and Radio Engineering (English translation of Elektrosvyaz and Radiotekhnika), 2019, 78(10), P. 869-878

23. V. Anavi-Chaput, K. Arrell, J. Baisden, R. Corrihons, D. Fallon, L. Siegmund, and N. Sokolof, “Planning for a Migration of PeopleSoft 7.5 from Oracle/UNIX to DB2 for OS/390,” Poughkeepsie, NY, USA, p. 148, 2000.

24. DataFlux, “The three key phases for data migration - and beyond,” New York, NY, USA, 2009.

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