Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)
Кафедра _____Системотехніки_____
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти _____другий (магістерський)_____

__Дослідження процесів міграції баз даних в корпоративних системах_____
_____
(тема)

Виконав:
студент _2__ курсу, групи ___ІТПм-18-1___
_____Кошова А.О._____
(прізвище, ініціали)

Спеціальність ___122 Комп'ютерні науки __
_____
(код і повна назва спеціальності)
Тип програми ____ освітньо-професійна ___
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційні технології проектування_____
(повна назва освітньої програми)

Керівник __проф.Гребеннік І.В_____
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Гребеннік І.В._____
(підпис) (прізвище, ініціали)

2019 р.

Харківський національний університет радіоелектроніки

Факультет *Комп'ютерних наук*

Кафедра *Системотехніки*

Рівень вищої освіти другий (магістерський)

Спеціальність 122 – Комп'ютерні науки та інформаційні технології

(код і повна назва)

Спеціалізація Інформаційні технології проектування

(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ

НА АТЕСТАЦІЙНУ РОБОТУ

Студентці Кошовій Алісі Олександровні

(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження процесів міграції баз даних в корпоративних системах

затверджена наказом по університету від "04" листопада 2019 р. № *1627Ст*

2. Термін подання студентом роботи 19 грудня 2019 р.

3. Вихідні дані до роботи *Функція: міграція даних між MS Server та PostgreSQL. Форма діалогу: інтерактивно-графічний режим. Перелік використаних програмних засобів: ОС Microsoft Windows 10, DBeaver, Технічне забезпечення: IBM–сумісний ПК Core i5 та вище.*

4. Зміст пояснювальної записки (перелік питань, що потрібно розробити) *4.1 Вступ 4.2 Аспекти міграції баз даних 4.3 Розробка методу міграції даних 4.4 Додаток для міграції 4.5 Висновки*

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслеників, плакатів) 5.1 *Практична міграція даних 5.2 Історія розвитку мови SQL 5.3 Процес міграції 5.4 Діаграма процесу міграції даних в нотації BPMN 5.5 Діаграма діяльності порівняння схем баз даних 5.6 Основна секція додатку з побудованим маршрутом* 5.*7 ER–діаграма словників для порівняння імен відносин і атрибутів*

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Спеціальна частина	*проф. Гребеннік І.В.*		
Розробка програмного засобу	*проф. Гребеннік І.В*		

7. Дата видачі завдання 04.11.2019

1 КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Отримання завдання на дипломне проектування*	*04.11.19*	
2.	*Аналіз завдання, літератури та аналогів згідно з темою дипломного проекту*	*04.11—10.11.19*	
3.	*Структурне проектування програмного засобу. Розробка алгоритму та структури програмного засобу*	*11.11—15.11.19*	
4.	*Вибір засобів автоматизації процесу проектування програмного засобу*	*15.11—20.11.19*	
5.	*Розробка проектних рішень по програмному та інформаційному забезпеченням програмного засобу*	*21.11—02.12.19*	
6.	*Проектування екранних форм та їх компонентів*	*03.12—07.12.19*	
7.	*Оформлення пояснювальної записки та програмної документації*	*08.12.—11.12.19*	
8.	*Оформлення графічної частини та презентаційних матеріалів комп'ютерного захисту*	*11.12—16.12.19*	
9.	*Представлення на рецензування*	*17.12.19*	
10.	*Представлення дипломного проекту в ЕК*	*19.12.19*	

Студент _____ Кошова А.О
(підпис)

Керівник роботи _____ *проф. Гребеннік І.В.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка до атестаційної роботи містить: 57 сторінок, 27 рисунків, 25 джерел.

БАЗИ ДАНИХ, SQL, МІГРАЦІЯ, МАСШТАБОВАННІСТЬ, MS SERVER, POSGRES, СХЕМА БАЗИ ДАНИХ, ЦІЛЬОВА БД.

Об'єктом дослідження є процесс міграції баз даних у корпоративних системах.

Предметом дослідження є методи та алгоритми міграції, а саме між СУБД MS Server та PostgreSQL.

Метою даної роботи є дослідження міграції баз даних та створення методу міграції даних, який має бути більш ефективним, ані ж методи, що існують та виконувати роботу оптимально.

В рамках даної роботи було проведено дослідження міграції між СУБД MS Server та PostgreSQL, була доведена актуальність дослідження, проаналізовані аналоги систем, що існують, обґрунтована мета розробки програмного засобу, поставлена задача дослідження.

На основі цих досліджень розроблений метод міграції баз даних.

В роботі досліджений новий метод міграції даних та проаналізована область використання нового методу.

ABSTRACT

Explanatory note to the certification work contains 57 pages, 27 figures, 25 sources.

DATABASE, SQL, MIGRATION, SCALING, MS SERVER, POSGRESQL, DATABASE SCHEME, TARGET DATABASE.

The purpose of this paper is to investigate database migration and to create a method of data migration that should be more efficient than existing methods and perform better.

In the framework of this work, the study of migration between DBMS MS Server and PostgreSQL was conducted, the relevance of the study was proved, the existing analogues of the system were analyzed, the purpose of software development was substantiated, the research task was set.

Based on these studies, a method for database migration has been developed.

The new method of data migration is explored and the area of application of the new method is analyzed.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних;

ІС – інформаційна система;

ІТ – інформаційні технології;

ОС – операційна система;

ПЗ – програмне забезпечення;

СУБД – Система Управління Базами Даних;

IDE – середовище розробки;

HTML – Hypertext Markup Language;

JSON – JavaScript Object Notation;

CSS – Cascade Style Sheets.

ВСТУП

Збільшення ролі інформаційних технологій в житті суспільства призводить до необхідності використання відповідних і зручних засобів для управління даними або інформацією. На сьогодні день існує широке поширення технологій реляційних баз даних, яке призвело до появи безлічі програмних продуктів з різними можливостями. Це ставить перед користувачем завдання вибору відповідної СУБД, яка може вирішити всі поставлені перед користувачем завдання. Міграція є методом переходу з однієї СУБД на іншу зі збереженням наявних даних.

Міграція баз даних - це досить складний процес, який має свої проблеми і ризики. У разі міграції промислової БД накладається ще більше умов і обмежень. В БД зберігається величезна кількість інформації, яка необхідна компанії для продовження бізнесу. Тому в результаті міграції в новій БД повинні міститися абсолютно всі дані. Крім того, БД є частиною великого бізнесу системи, вона повинна взаємодіяти з іншими елементами, обмінюючись з ними даними. Після міграції всі інші додатки повинні отримувати коректні дані від нової БД. Також час на проведення міграції сильно обмежена, оскільки система повинна працювати практично 24 години 7 днів на тиждень. Міграцію можна проводити тільки протягом невеликого технічної перерви. В результаті необхідно відразу отримати систему, яка працює, що містить всі дані, що зберігалися в вихідної БД, щоб після закінчення процесу користувачі системи могли продовжити роботу.

Основна мета проекту міграції - переклад вихідної бази даних з однієї бази в абсолютно ідентичну, але іншого вендору. До результату міграції ставляться такі вимоги.

- Повинні бути перенесені всі дані з вихідної бази.
- Повинна бути збережена функціональність вихідної бази.

Також є додаткове нефункціональне і, на перший погляд, не формалізується вимога мінімізації змін в схемі даних і в сигнатури збережених процедур для спрощення подальшої адаптації реальних клієнтських додатків до нової бази даних.

У даній роботі будуть розглянуті такі СУБД, як Microsoft SQL Server і PostgreSQL. Для написання функцій в PostgreSQL використовується вбудовану процедурну мову PL / pqSQL, можливості якого багато в чому аналогічні Transact-SQL, який використовується в СУБД MS SQL Server. Але існує ряд відмінностей між цими мовами. Ряд конструкцій Transact-SQL не підтримує Postgres, а деякі мають альтернативу. Метою даного дипломного проекту буде реалізація програми, що перетворювала би компоненти бази даних MS SQL Server (таблиці, уявлення, процедури, функції) в код, що справно працює на PostgreSQL, що має той же функціонал. При конвертації компонентів БД виникають такі проблеми:

- відмінність в засобах СУБД MS SQL Server і PostgreSQL;
- відсутність збережених процедур в PostgreSQL;
- різні типи даних в MS SQL Server і PostgreSQL;

Серед основних завдань будуть розглядатися наступні завдання:

- аналіз проблеми міграції компонентів бази даних;
- аналіз аналогів;
- формулювання завдання конвертації;
- аналіз відмінностей між засобами СУБД MS SQL Server і PostgreSQL
- розробка алгоритму роботи ПО для конвертації;
- розробка програмного забезпечення міграції;
- тестування додатка для міграцій;

Крім вирішення основних завдань, можуть виникнути додаткові, другорядні завдання, які дозволять остаточно спроектувати додаток для міграції компонентів бази даних MS SQL Server.

1 АСПЕКТИ МІГРАЦІЇ БАЗ ДАНИХ

1.1 Розуміння міграції 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.

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. [1]

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. Запис може бути перенесена тільки в тому випадку, якщо вона є незалежною, або записи, від яких залежить ця, 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 реалізується у вигляді процедури, яка спочатку викликається для незалежних записів, а далі рекурсивно і для залежних від цього запису. Тому порядок пересування між 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.

Найпростішим варіантом є створення проміжної програми. Вона повинна зв'язуватися з вихідною і цільовою базами даних і виконувати необхідні перетворення.

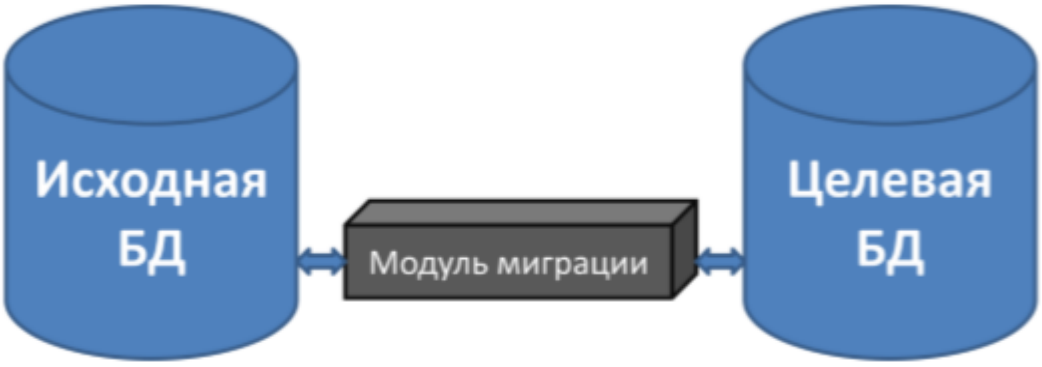


Рисунок 1.1 – Варіант міграції даних з проміжною програмою

Інший варіант міграції даних (цей варіант більш кращий при переході на нову систему управління базами даних (далі, СУБД)) – попереднє завантаження старих даних в тимчасові таблиці нової СУБД. Сучасні СУБД (наприклад, Oracle)[2] зазвичай містять спеціальні утиліти, які дозволяють виробляти дуже швидке завантаження зовнішніх даних різних форматів. В такому випадку модуль міграції пишеться засобами процедурних мов, вбудованих в СУБД (наприклад, PL / SQL або java).

Рисунок 1.2 – Варіант міграції даних засобами СУБД.

Міграція бази даних – в контексті корпоративних програм – означає переміщення ваших даних з однієї платформи на іншу. Є багато причин, з яких ви можете перейти на іншу платформу. Наприклад, компанія може вирішити заощадити гроші, перемістившись у хмарну базу даних. Або компанія може виявити, що якесь певне програмне забезпечення баз даних має функції, що є критичними для їхніх бізнес–потреб. Або застарілі системи просто застаріли. Процес міграції баз даних може містити в собі кілька етапів та ітерацій – включаючи оцінку поточних баз даних та майбутніх потреб компанії, міграцію схеми та нормалізацію та переміщення даних. Плюс тестування, тестування та багато іншого.

Недоліки міграції баз даних – це витрати. Однією з головних причин того, що компанії мігрують бази даних, є економія коштів. Часто компанії переходять від локальної бази даних до хмарної бази даних. Це економить на інфраструктурі, а також на робочій силі та досвіді, необхідних для її підтримки.[3]

Модернізоване програмне забезпечення. Ще одна поширена причина міграції – перехід від застарілої системи чи застарілих систем до системи, розробленої для сучасних потреб у даних. В епоху великих даних нові методи зберігання є необхідністю. Наприклад, компанія може вибрати перехід зі застарілої бази даних SQL до озера даних або іншої гнучкої системи.

Одне джерело істини. Ще одна поширена причина міграції даних – переміщення всіх даних в одне місце, доступне всім підрозділам компанії. 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.

1.2 Проблеми з міграцією баз даних

Міграція баз даних може бути дуже складною, але при правильному плануванні ці поширені проблеми можна пом'якшити.

Існують такі проблеми з міграцією:

– пошук баз даних.;

– втрата даних або пошкодження;

– безпека

Проблема №1: пошук баз даних. Якщо компанія існує більше 5 років, ймовірно, вона має багато різних баз даних, які існують у різних частинах компанії. Вони можуть бути в різних відділах і різних країнах. Вони, можливо, були залучені через придбання. Частина завдання з міграції баз даних полягає у пошуку розрізнених баз даних у компанії та плануванні способів нормалізації даних та перетворення схем.

Проблема №2: втрата даних або пошкодження. Під час міграції баз даних важливо забезпечити безпечне переміщення ваших даних без втрат чи пошкоджень. Потрібно буде спланувати, як перевірити втрату даних або пошкодження, які можуть статися при переміщенні даних з однієї системи в іншу.

Проблема №3: безпека Переміщуючи дані з однієї платформи на іншу, важливо захистити дані. На жаль, є багато недобрих акторів, які хотіли б отримати свої особисті дані, які ви зберігаєте. Можна вибрати шифрування даних або видалити особисту інформацію (PII) як частину міграційного процесу.

Міграція бази даних – це багатофазний процес, який включає деякі або всі наступні етапи:

- Оцінка
- Перетворення схеми бази даних
- Міграція даних
- Тестування та налаштування

Оцінка. На цьому етапі потрібно буде зібрати вимоги бізнесу, оцінити витрати та вигоди та виконати профілювання даних. Профілювання даних – це процес, за допомогою якого ви можете ознайомитись схемою даних, що існує та бази даних. Також потрібно буде спланувати спосіб переміщення даних – чи будете використовувати інструмент ETL (Видобуток, Перетворення та Завантаження), сценарій чи інший інструмент для переміщення даних.

Перетворення схеми бази даних. Схема – це креслення структури бази даних, і вона змінюється, виходячи з правил даної бази даних. При переміщенні даних з однієї системи в іншу вам потрібно буде конвертувати схеми так, щоб структура даних працювала з новою базою даних.

Міграція даних. Після того, як ви виконали всі попередні вимоги, вам потрібно буде фактично перемістити дані. Це може включати сценарії, використовуючи інструмент ETL або інший інструмент для переміщення даних. Під час міграції ви, ймовірно, перетворите дані, нормалізуєте типи даних та перевірите наявність помилок.

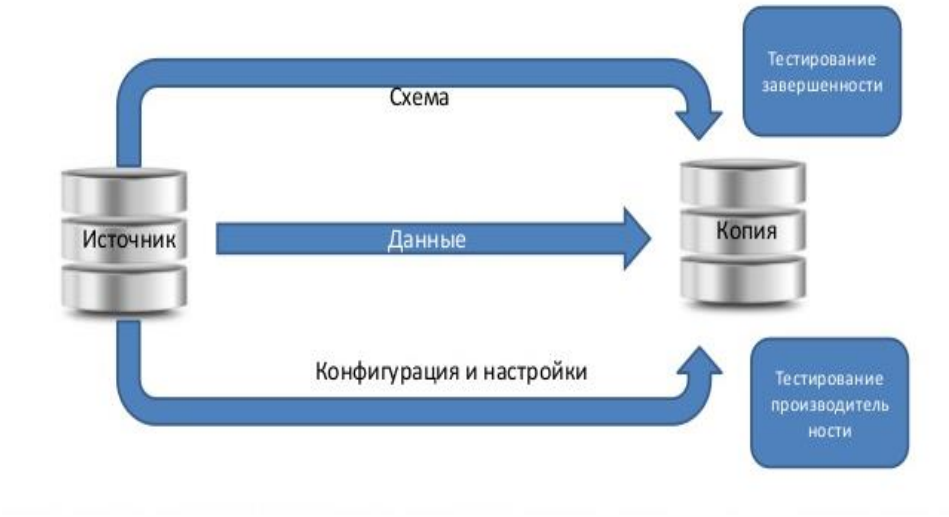


Рисунок 1.3 – Практична міграція даних

Тестування та налаштування. Після переміщення даних вам потрібно переконатися, що дані: переміщено правильно, є завершеним, не має відсутніх значень, не містить нульових значень і є дійсним. Схеми обох баз даних та самі данні повинні бути ідентичними.[4,5]

1.3. Аналіз представлених на ринку продуктів міграції даних

Міграція дуже часто потрібна в корпоративних системах, якщо компанія здійснює перехід від одної технології і іншу, або взагалі минула база устаріла. Тому міграція не є інновацією, існує декілька варіантів додатків для міграції, але вони дуже дорого обходяться маленьким компаніям.

Далі представлений список аналогів системи:

1) Migration Architect

Виробник: Evoke software. Дана система призначена для планування і здійснення наступних проектів: створення інформаційних сховищ і вітрин даних;

міграція даних в готові програми; міграція даних і додатків, пов'язана з переходом на нову СУБД; міграція даних при зміні схеми БД. Основні можливості:

- Аналіз не тільки метаданих, а й фактичних даних для більш точного визначення області значень атрибутів і перевірки наявності первинних ключів.
- Визначення відносин між атрибутами (функціональні залежності) і відносин між таблицями (виявлення надмірності даних).
- Підтримка простої моделі для збереження і визначення відображень між атрибутами вихідної і цільової схеми БД.[6]

2) TRUmigrate

Виробник: Valiance partners, http://www.valiancepartners.com

Основні можливості:

- Завдання правил. Відображення задаються засобами графічного призначеного для користувача інтерфейсу і зберігаються в XML–файлі.
- Експорт. Витяг вмісту та мета інформації з одного або декількох сховищ даних і перенесення отриманих даних в проміжне сховище.
- Доповнення та об'єднання. Додавання даних з допоміжних джерел і об'єднання з раніше отриманими даними.
- Перетворення. Застосування правил перетворень для отриманих даних.
- Імпорт. Завантаження вмісту об'єктів в цільовий додаток і ініціалізація з параметрами налаштування.[7]

2) AWS Database Migration Service and Amazon DynamoDB

Багато організацій вважають перехід від комерційних реляційних баз даних, таких як Microsoft SQL Server або Oracle, до Amazon DynamoDB – повністю керованого, швидкого, високо масштабного та гнучкого сервісу баз даних NoSQL. Amazon DynamoDB може збільшити або зменшити пропускну спроможність на основі трафіку відповідно до потреб бізнесу. Це повністю керована хмарна база даних, яка підтримує як моделі зберігання документів, так і ключові значення. Його гнучка модель даних, надійні показники та автоматичне масштабування

пропускної спроможності роблять її прекрасною для мобільних пристроїв, Інтернету, ігор, рекламних технологій, IoT та багатьох інших програм.

Переваги AWS Database Migration Service:

- Простота використання:
- Мінімальні простої;
- Підтримка поширених баз даних;

Сервіс AWS Database Migration Service простий у використанні. Не потрібно встановлювати будь-які драйвери або додатки, і в більшості випадків не потрібно вносити зміни в вихідну базу даних.

AWS Database Migration Service дозволяє здійснити міграцію баз даних на платформу AWS практично без простоїв. Все зміни, що вносяться до вихідну базу даних під час міграції, безперервно реплицируются в цільову базу даних, при цьому вихідна база даних залишається в повністю робочому стані. AWS Database Migration Service дозволяє виконувати міграцію даних, використовуючи в якості вихідної і цільової баз більшість поширених баз даних, як комерційних, так і з відкритим вихідним кодом. Сервіс підтримує як однорідні міграції, наприклад з Oracle в Oracle, так і різнорідні міграції між різними платформами баз даних, наприклад з Oracle в Amazon Aurora.

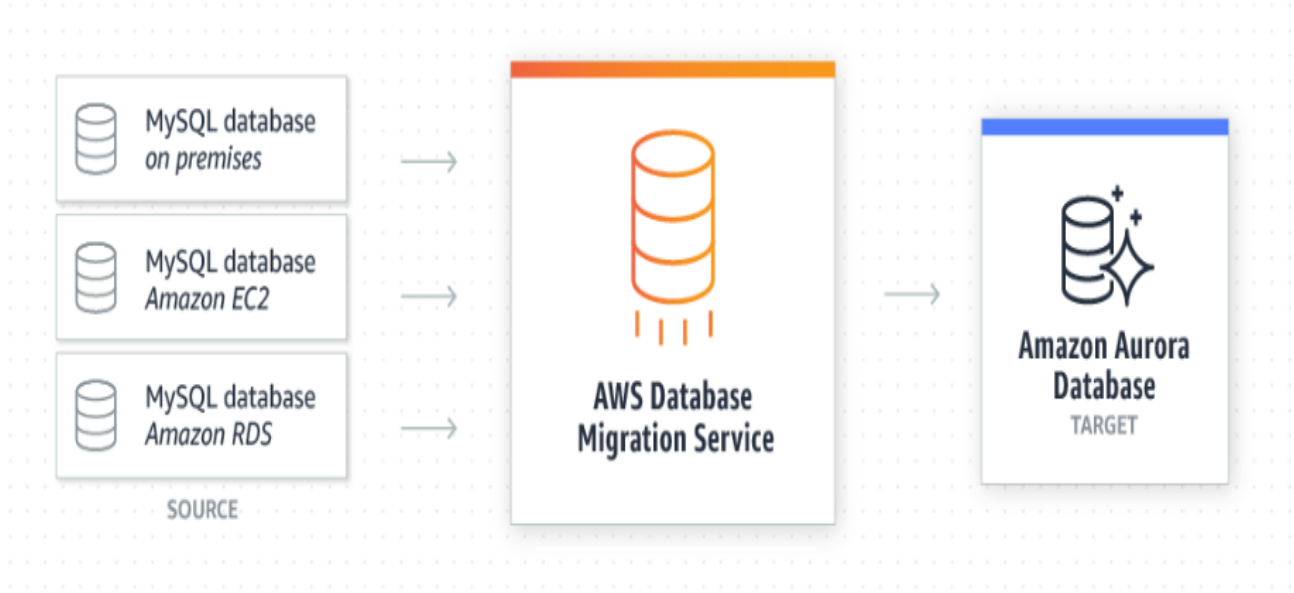


Рисунок 1.4 – AWS Database Migration Service.

1.4. Види міграції даних

Використання міграції даних для досягнення різних цілей призвело до формування окремих видів міграції даних:

1. за кількістю виконуваних сценаріїв в одиницю часу [9]:
    - послідовна;
    - паралельна.

Пояснення цих видів міграції наведено на прикладі реляційних баз даних. При послідовної міграції відбувається виконання одного сценарію в одиницю часу. Порядок виконання сценаріїв заснований на залежності між відносинами. При паралельної міграції даних виконується більш детальний аналіз залежностей між відносинами. Виконання сценаріїв в цьому випадку відбувається в той момент, коли дані всіх необхідних відносин вже перенесені.

2. за типом сховищ даних:
    - між сховищами одного типу [12, 19, 25];
    - між сховищами різних типів [1, 22, 24].

Сховища даних — основа для побудови систем підтримки прийняття рішень. Основна мета створення сховища в тому, щоб зробити усі значимі для управління бізнесом дані доступними в стандартизованій формі, придатними для аналізу та отримання необхідних звітів.[26]

Міграція однорідних баз даних. При міграції однорідних баз даних ядра вихідної і цільової бази даних однакові або сумісні між собою, наприклад Oracle і Amazon RDS для Oracle, MySQL і Amazon Aurora, MySQL і Amazon RDS для

MySQL або Microsoft SQL Server і Amazon RDS для SQL Server. Оскільки структури схем, типи даних і коди вихідної і цільової баз даних сумісні, така міграція виконується за один крок. Ви створюєте завдання міграції, що визначає підключення до вихідної і цільової баз даних, і запускаєте міграцію одним натисканням кнопки. Все інше виконує сервіс AWS Database Migration Service. Вихідна база даних може знаходитися у вашій локальній мережі, поза AWS, працювати в інстанси Amazon EC2 або бути базою даних Amazon RDS. Цільовий базою даних може бути база даних в Amazon EC2 або Amazon RDS.

Міграція різнорідних баз даних. При міграції різнорідних баз даних ядра вихідної і цільової баз даних відрізняються. Це може бути міграція з Oracle в Amazon Aurora, з Oracle в PostgreSQL або з Microsoft SQL Server в MySQL. В цьому випадку структури схем, типи даних і коди вихідної і цільової баз даних сильно відрізняються, і перед міграцією даних необхідно виконати перетворення схеми і коду бази даних. Тому міграція різнорідних баз даних виконується в два етапи. Спочатку використовується інструмент AWS Schema Conversion Tool для конвертації схеми і коду вихідної бази даних у відповідну схему і код цільової бази даних, а потім за допомогою сервісу AWS Database Migration Service виконується міграція даних з вихідної бази даних в цільову. Необхідне перетворення типів даних автоматично виконується сервісом AWS Database Migration Service під час міграції. Вихідна база даних може знаходитися у вашій локальній мережі, поза AWS, працювати в інстанси Amazon EC2 або бути базою даних Amazon RDS. Цільовий базою даних може бути база даних в Amazon EC2 або Amazon RDS.

1.5. Загальні принципи виконання міграції даних

Успішність виконання міграції даних залежить від багатьох факторів. Вплив технологічних, людських чинників, а також факторів, пов'язаних з даними. Прикладами технологічних факторів є обсяг їх даних, що переносяться і частота

виконання їх перенесення. До факторів, пов'язаних з даними, відносять якість даних, схожість вихідної і цільової структур даних, залежності між даними і інші. Як будь–які IT проекти, проекти з міграції даних схильні до ризиків. Постійно зростаючий попит на ефективні IT–рішення, швидко зростаюча пропозиція нових технологій та високий рівень конкуренції в IT–секторі значно підвищили потребу в міграції даних з одного середовища в інше. З частими передачами даних також зростала кількість помилок. Брак досвіду та знань є основною причиною головних болів при перенесенні даних. Найпоширеніші проблеми компанії:

- Бюджет;
- Простота;
- Безпека процесу;
- Сумістніть середовищ;
- Досвід працівників;
- Поведінка бізнес–процесів в нових умовах;
- Час;
- Тестування;

Дуже часто міграційні проекти не спрацьовують, коли їх впроваджують, оскільки непрофесійний або необережний план міграції призводить до додаткових витрат або навіть втрат для бізнесу, коли простоює. Міграцію даних слід розглядати як інвестицію, яка потрібна одночасно з впровадженням нової системи IT. Не планувати міграційні витрати при впровадженні нової IT–платформи – це як не планувати витрати на пальне при покупці нового автомобіля.

Дані самі по собі відіграють значну роль у міграції. Від того, як дані інтегруються в бізнес, як вони змінюються з часом, певні набори даних стають взаємозалежними, що ускладнює їх перенесення в нове середовище. Дуже важливо вивчити дані, додатки та компоненти системи, що використовуються для складності та взаємозалежності, для поліпшення управління даними та мобільності додатків. Оскільки майже всі дані інтегровані в додатки, їх рівень інтеграції є важливою складовою процесу міграції.

Успішна міграція даних ґрунтується на точних деталях, визначених у плані міграції. Це не тільки план, який описує, що робити, але також, коли робити і що робити у випадку помилки чи проблеми. Продуманий і точний план вимагає додаткового часу, досвіду та досвіду. План визначає, який метод буде використовуватися для імпорту та експорту даних, відновлення мережі та підготовки ресурсів.

Одним із способів зниження ризиків для проектів з міграції даних є розбиття на фази або етапи. Так, в існує процес міграції даних, що складається з 7 фазах: стратегія, аналіз, проектування, складання, тестування [17] / впровадження, перевірка, супровід, а іноді всього три етапи – аналіз, тестування та виконання [8]. На прикладі міграції з СУБД Oracle в СУБД Microsoft SQL Server при використанні процесу міграції баз даних, що складається з 7 кроків, в [12] описані можливості застосування програмних продуктів компанії DB Best на кожному з кроків і наведені частки виконання завдань процесу міграції даних у відсотках при використанні описаних програмних продуктів. Такі розбиття дозволяють поліпшити управління проектом, а також вибрати відповідні програмні продукти для виконання окремих завдань або процесу міграції даних в цілому. У літературі вказуються характеристики, якими повинні володіти додатки для перенесення даних: протоколювання, повідомлення про помилки.

Важливою частиною процесу міграції даних є тестування. Воно дозволяє гарантувати, що в цільовому сховищі даних немає дефектів, сховище містить всі необхідні дані, дані перенесені в необхідні структури і т. Д. Не дивлячись на важливість цього аспекту, тестування залишається мало освітленим при описі методів і підходів до міграції даних [8, 12, 17]. В [4] описаний життєвий цикл тестування міграції даних, огляд методів тестування і контролю якості наведено в [27].

Основою механізму контролю якості процесу перевантаження даних є аналіз повертаючих кодів та журналів усіх утиліт, що беруть участь у процесі перевантаження, моніторинг цілісності інструментами БД та додатковий контроль результату перевантаження – перевірка. Перевірка дозволяє переконатися, що всі

дані успішно перевантажені в нову базу даних, зазнавши необхідних документально підтверджених змін. Доречність перевірки під час міграції баз даних такого обсягу важко недооцінити. По–перше, у схемі є понад 2 тис. Таблиць і майже 10 000 стовпців, а деякі таблиці містять десятки мільйонів записів. По–друге, іноземні ключі, як засіб контролю цілісності, практично не використовуються в оригінальній базі даних. Процес перевірки повинен впливати не тільки на вміст об'єктів, але і на всю схему даних, перевіряючи наявність та стан об'єктів. Повна перевірка для встановлення ідентичності двох баз такого розміру вимагає великих ресурсів, особливо часу. Тому до процесу потрібен підхід, який вимагає значно менших ресурсів і в той же час володіє високим ступенем надійності результату.

1.6. Підзадачі міграції даних

Відомі підходи і методи міграції даних явним або неявним чином виконують аналіз сховищ даних і перенесення даних.

В рамках аналізу сховищ даних виконується порівняння або зіставлення структур вихідних і цільових сховищ даних. Під зіставленням схем (schema matching) розуміють завдання визначення семантичних відповідностей між елементами схем [2]. Зіставлення схем традиційно вважається AI–повної завданням [3], і це накладає обмеження на варіанти її вирішення. Класифікації підходів до зіставлення схем можна знайти в оглядах [5,12]. У літературі зустрічається використання терміна «порівняння схем» 2 як синонім для зіставлення схем.

Ідеї пропонованих методів і підходів до зіставлення схем ґрунтуються на використанні зіставлення графів [2], теорії ймовірності [3], машинного навчання

[3] та інших областей [21]. Найбільш важливими характеристиками методів і підходів до зіставлення схем є точність результату і трудомісткість [23]. Оскільки AI–повнота зіставлення схем обумовлює необхідність коригування результатів зіставлення людиною, а великі розміри схем збільшують час виконання зіставлення, велику увагу приділяють занадто багато роботи пропонованих алгоритмів. Наприклад, при використанні зіставлення графів слід враховувати, що в загальному випадку задача зіставлення графів не віднесено ні до класу P, ні до класу NP– повних задач, а завдання ізоморфізму підграфу (один з типів зіставлення графів [7]) є NP–повною [4]. Таким чином, при необхідності зіставлення схем вибір методу для його виконання слід проводити, керуючись допустимими співвідношеннями точності і трудомісткості.

При міграції даних між сховищами даних різних типів спостановлення структур ускладнюється використанням різних моделей даних. Наприклад, в [24] розглядається міграція реляційних баз даних в об'єктно–орієнтовані або XML бази даних. Складність полягає в тому, що найчастіше немає готових і / або однозначних правил порівняння елементів різних моделей даних. Можливо кілька підходів до зіставлення елементів різних моделей даних. Один з таких підходів полягає у визначенні правил порівняння для кожної можливої пари моделей даних. Інший підхід ґрунтується на використанні проміжного концептуального уявлення, в якому будуть представлені вихідні і цільові структури даних. Так, в якості проміжного представлення пропонуються граф [2, 3], реляційна модель [25], семантичні моделі [5], XML файли [8]. Мінуси такого підходу полягають у можливій втраті семантики при переході до проміжного поданням, а також у збільшенні трудомісткості. Очевидно, що міграція даних між сховищами даних одного типу не стикається з вищеописаними проблемами, але при цьому обмежується область її застосування. Однак навіть у разі міграції даних між сховищами даних одного типу існують складності при виконанні порівняння структур даних. Вони пов'язані з відмінностями конкретних реалізацій сховищ даних [12, 2]. Наприклад, в одному сховищі даних довжина даних типу «рядок»

обмежена, а іншому – ні; в реляційних СУБД можуть використовуватися різні стандарти мови SQL.

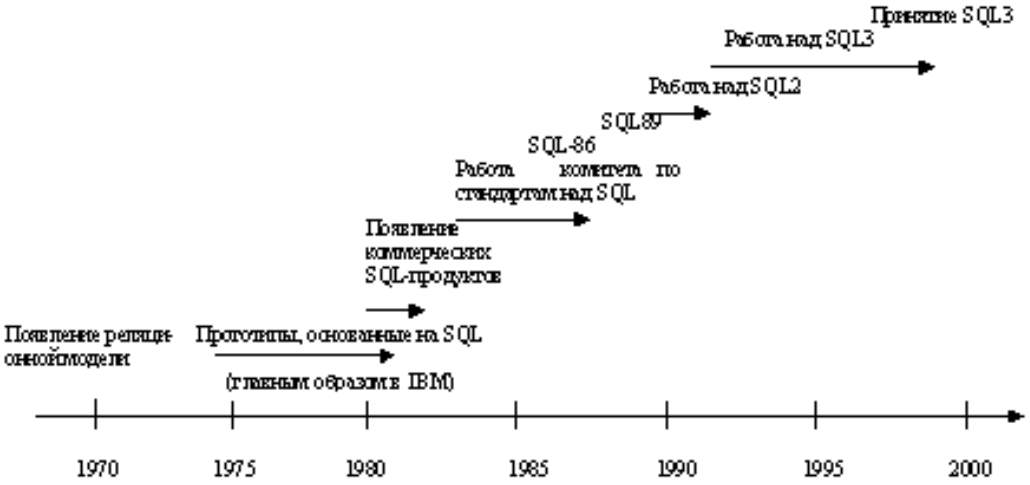


Рисунок 1.5 – Історія розвитку мови SQL

Різні уявлення об'єктів моделюється світу і зв'язків між ними також ускладнюють порівняння структур даних вихідних і цільових сховищ даних. Говорячи термінами ER–моделі Чена [53], одне й те саме явище, що моделюється світу може бути представлено як атрибут, як безліч сутностей і як безліч зв'язків. Таким чином, з'являється проблема зіставлення об'єктів і їх зв'язків в силу можливості використання різних їх уявлень. До цієї проблеми також стосується використання еквівалентних конструкцій моделей даних і різних точок зору при поданні об'єктів моделюється світу, наприклад, використання різних імен (синонімів, гиперонімів), обмежень цілісності, типів даних. Класифікації можливих відмінностей між уявленнями об'єктів і їх зв'язків наведені в [5, 21].

Здійснення перенесення даних пов'язано з рішенням деяких питань. Одним з них є визначення порядку перенесення даних [52]. Розв'язання даного питання залежить від наявності зв'язків між об'єктами моделеваного світу, представленими в сховище даних. У разі, коли в сховище даних представлений лише один тип об'єктів моделюється світу без зв'язків (наприклад, для реляційної

моделі це означає, що є тільки одне відношення, яке не має зовнішніх ключів), складнощів у визначенні порядку перенесення даних немає. Випадок, коли в сховище даних представлено кілька типів об'єктів без зв'язків, також не приносить особливих труднощів: перенесення виконується для кожного типу об'єктів в будь–якому порядку.

1.7. Процес міграції даних

Пропонований процес орієнтований на виконання таких сценаріїв для міграції даних, що зберігаються в реляційних базах даних: перенесення даних в БД зі схемою, зміненої внаслідок модифікації на замовлення користувача або зміни версії програми, що використовує базу даних;

- перехід на іншу версію СУБД;
- перехід на іншу реляційну СУБД;
- різні комбінації наведених вище пунктів.

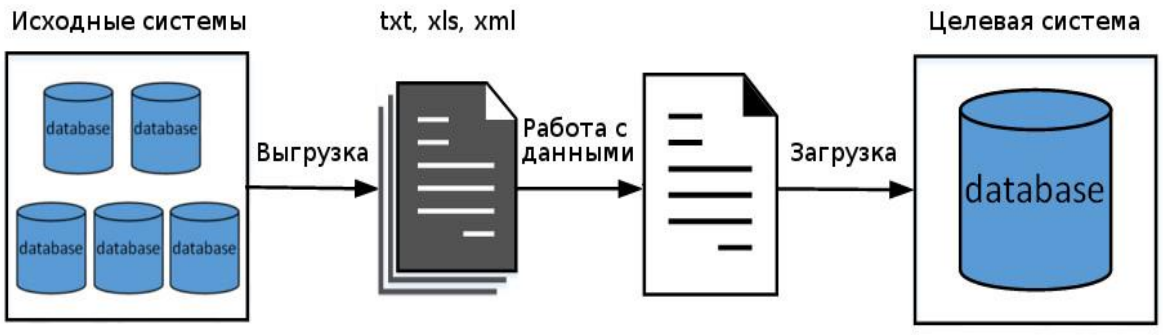


Рисунок 1.6– Процесс міграції у простому вигляді

Пропонований процес міграції даних можна проілюструвати діаграмою в нотації BPMN, наведеної на рисунку 3.

Етапами такого процесу міграції даних є:

- витяг схем вихідної і цільової баз даних;
- побудова послідовностей обходу схем БД;
- порівняння схем баз даних;
- завдання правил перенесення даних;
- перенесення даних.

Виконання першого етапу має на увазі уявлення схем баз даних в оперативній пам'яті у вигляді, що містить необхідні для перенесення даних відомості і приховує деталі роботи з конкретними СУБД від інших кроків процесу міграції даних. Ця інформація потрібна протягом усього процесу, тому неуспішне її витяг призводить до завершення процесу.

Другий етап вирішує завдання коректного перенесення даних з вихідної бази даних, а також формує інформацію, необхідну для наступних етапів. Під послідовністю обходу схеми БД розуміється послідовність відносин, сформована за принципом топологічної сортування орієнтованого графа з деякими особливостями. Така послідовність будується для забезпечення збереження посилальної цілісності при перенесенні даних, а також використовується при виконанні порівняння схем баз даних.

Порівняння схем в пропонованому процесі міграції даних є допоміжним етапом, результат якого буде підказкою для людини при завданні правил переносу – алгоритмічних дій над кортежами з відносин, що визначають як, куди і які дані переносити.

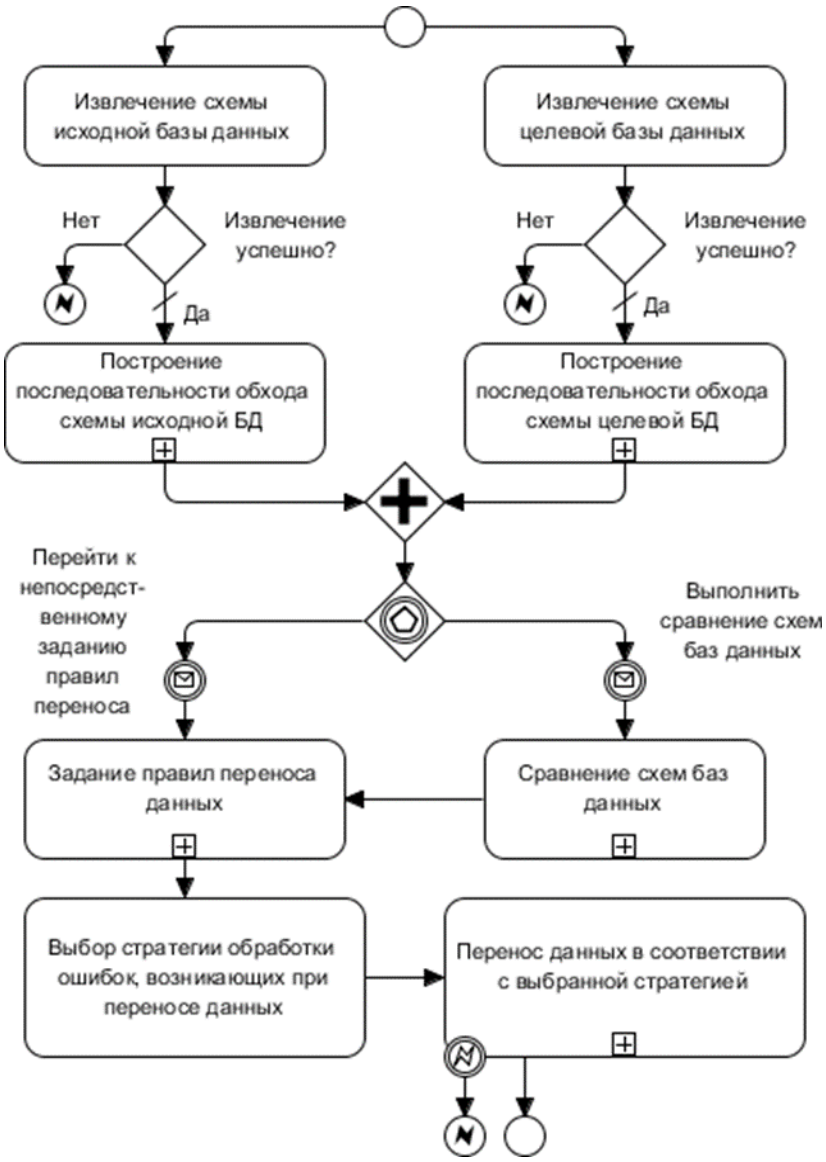


Рисунок 1.7– Діаграма процесу міграції даних в нотації BPMN

Рисунок 1.8 – Схема вихідної бази даних

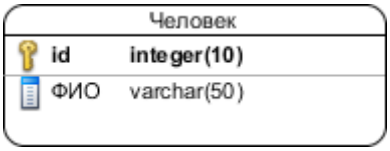


Рисунок 1.9– Схема цільової бази даних

На формування правил переносу даних впливають в першу чергу відмінності між схемами вихідної і цільової баз даних. Такий випадок ілюструють рисунки 2 і 3. Проте варто зауважити, що існують правила перенесення даних, пов'язані з перетвореннями самих даних незалежно від відмінностей між схемами баз даних. Прикладом такого правила є формування одного кортежу в цільової базі даних з декількох кортежів з вихідної бази даних при повному збігу схем. Передбачається, що правила перенесення будуть записуватися у вигляді сценаріїв і для завдання правил перетворення в зумовлених випадках, наприклад, при перейменуванні атрибута, будуть передбачені шаблони сценаріїв.

На етапі завдання правил також здійснюється настройка виконання перенесення даних вибором однієї з можливих стратегій обробки помилок:

- зупинка при першій помилку – відбувається завершення процесу міграції при першій помилці;
- дозвіл помилок користувачем – при виникненні помилки користувачеві пропонується скорегувати дані, після чого міграція даних триває, починаючи з спроби перенесення даних, на яких сталася помилка;
- пропуск кортежів з помилками – при виникненні помилки пропускаються кортежі з некоректними даними та всіх залежних від них кортежі; міграція даних триває;

заповнення значеннями за замовчуванням – в цьому випадку відбувається коригування кортежів з помилковими даними за допомогою значень за замовчуванням; міграція даних триває з спроби перенесення відкоригованих даних.

Прикладом помилки перенесення даних є перетворення в число рядка, що не представляє число.

На останньому етапі відбувається перенесення даних з вихідної бази даних в цільову, ідея якого полягає в понятті залежності кортежів. Незалежним кортежем називається кортеж з відношеннями, що не має зовнішніх ключів, або кортеж з відношеннями, що має зовнішні ключі, значення яких для даного кортежу

невизначені (NULL). В іншому випадку кортеж називають залежним. Перенесення даних здійснюється згідно з наступним принципом: спочатку переносяться незалежні кортежі, а потім залежні кортежі. Варто зауважити, що залежний кортеж може бути перенесений тільки тоді, коли всі кортежі, від яких він залежить вже перенесені. Механізм переносу даних крім сценаріїв перетворення, які задають правила перенесення, використовує ще два види сценаріїв. Перший вид сценаріїв служить для відключення деяких обмежень цілісності в цільової базі даних, при наявності яких перенесення даних не здійснимо, а другий – для їх включення після перенесення даних.

1.8 Постановка задачі

У рамках атестаційної роботи необхідно дослідити процеси та механізми міграції даних та реалізувати програмний продукт, що виконує міграцію даних з бази даних MS Server на PostgreSQL.

Серед завдань можна виділити наступні:

1. Отримання метаінформації про схеми баз даних
2. Порівняння схем
3. Побудова послідовності обходу схеми вихідної БД
4. Перенесення даних з перетвореннями – на льоту.

В рамках даної роботи вирішуються останні три завдання, а саме:

- Провести аналіз різних топологій схем баз даних і закріплення правил обходу схеми БД для конкретної топології.

- На основі отриманих правил розробити алгоритм побудови послідовності обходу схеми вихідної БД, який надалі задасть порядок, згідно з яким буде здійснюватися перенесення даних з вихідної БД в цільову БД.
- Розробити механізм перенесення даних.

У рамках даної роботи необхідно дослідити механізми міграції даних та реалізувати програмний продукт, що виробляє міграцію даних.

На даний момент не існує єдиного підходу до вирішення проблеми міграції даних. Фірма, яка розробляє програмне забезпечення, не завжди підтримує можливість збереження накопичених даних при переході на більш пізню версію програмного продукту. За прохання клієнта дана проблема вирішується в індивідуальному порядку, і, як правило, послуга коштує чималих грошей. Розробникам доводиться аналізувати всі зміни та кожен раз створювати нові процедури перенесення даних для конкретного випадку зміни схеми БД. При цьому кожна фірма дотримується своєї політики вирішення даного завдання, яка найчастіше не розголошується. Тому постає питання про вивчення проблеми міграції даних, що зберігаються в реляційних СУБД і написанні відповідного програмного продукту.

Передбачається, що даний програмний продукт буде проводити автоматичний аналіз змін схем БД і генерувати правила перенесення, надаючи користувачу можливість вибору альтернатив при виникненні спірних ситуацій і помилок.

2 РОЗРОБКА МЕТОДУ МІГРАЦІЇ ДАНИХ

2.1 Перехід від послідовної міграції даних до паралельної

Важливим критерієм для міграції даних є час її виконання. Одним із способів його скорочення є перехід від послідовної міграції даних до паралельної.

Для виконання паралельної міграції даних відповідно до описаного раніше процесу необхідно виділити незалежні частини даних. Для цього пропонується спочатку виділити компоненти зв'язності на графі, що представляє схему бази даних, а потім для кожної компоненти зв'язності побудувати послідовність обходу. Виділені компоненти зв'язності можна переносити паралельно. Однак ситуація, коли в базах даних є такі незалежні частини, буває досить рідко, і, отже, скорочення часу виконання міграції даних за допомогою такого розпаралелювання буде в рідкісних випадках. У зв'язку з цим пропонується ввести розпаралелювання всередині компонент зв'язності.

- якщо є вершини, полустепені заходу яких дорівнює 0, вибрати все такі вершини;
- якщо таких вершин немає, вибрати будь–яку вершину з найменшою полустепені заходу і з найбільшою полустепені результату серед вершин, що представляють відношення з обмеженнями посилальної цілісності, що допускають невизначені значення;
- інакше вибрати будь–яку вершину серед вершин з найменшою полустепені заходу і з найбільшою полустепені результату.

Паралельний перенос усередині однієї компоненти зв'язності почнеться з паралельного перенесення кортежів вершин з найменшим порядковим номером кроку побудови. Потім після завершення перенесення даних чергової вершини для подальшого перенесення будуть вибиратися вершини з наступним порядковим номером кроку, якщо дані всіх вершин, від яких вони залежать, вже

перенесені. Для збереження посилальної цілісності необхідно проміжне сховище для даних про унікальні ідентифікатори перенесених кортежів. Видалення частини інформації з проміжного сховища в процесі перенесення даних можливо за наступним принципом: дані про кортежі можуть бути видалені після того, як перенесені всі залежні від них кортежі. Таким чином, описаний механізм паралельної міграції даних дозволяє скоротити час виконання в порівнянні з послідовною міграцією даних у випадках, коли вихідна і цільова бази даних не накладають обмеження на можливість паралельного перенесення. Використання проміжного сховища дозволяє також вирішити проблему, що з'являється при необхідності зміни значень унікальних ідентифікаторів, і забезпечити можливість відкоту і «гарячого старту» міграції даних.

2.2 Синтаксичні угоди

Головне в процесі міграції з однієї бази на іншу – це детальне вивчення синтасисів та відмінностей у реалізації основних об’єктів схеми бази даних

2.2.1 Таблиці та стовпці

Іноді назви таблиць або стовпців MS SQL містяться у квадратних дужках у запитах (наприклад, якщо вони містять пробіли або з інших причин). PostgreSQL не дозволяє квадратні дужки навколо таблиці імен стовпців, всі вони повинні бути замінені на " [object] –>" object ".

MS SQL

```
SELECT [id]                          -- Delimiter is optional

  FROM [dbo].[Conventions test] -- Identifier contains a space
and uses a reserved keyword.

 WHERE [order] IS NULL;          -- Identifier is a reserved
keyword.
```

Рисунок 2.1 – Приклад фрагменту мови MS SQL.

PostgreSQL

```
SELECT id               /* Delimiter is optional */
  FROM gold_test_ss_dbo."Conventions test" /* Identifier
contains a space and uses a reserved keyword. */
 WHERE "order" IS NULL;   /* Identifier is a reserved keyword.
*/
```

Рисунок 2.2.– Приклад фрагменту мови PostgreSQL.

2.2.2 Схеми

Повна назва об'єкта складається з чотирьох ідентифікаторів: ім'я сервера, ім'я бази даних, ім'я схеми та ім'я об'єкта. Вони відображаються у такому форматі:

```
server_name.[database_name].[schema_name].object_name
| database_name.[schema_name].object_name
| schema_name.object_name
| object_name
```

Рисунок 2.3 – Повна назва об'єкта

При використанні схеми повне ім'я об'єкта бази даних у запиті може виглядати як database.schema.object. При переході до PostgreSQL всі назви схем та назви даних повинні бути перетворені в імена схем, що складаються з імені_бази_ ім'я_схеми. Якщо вихідний код не має імен бази даних та імен схем, вам потрібно в будь–якому випадку написати schema_name перед назвою об'єктів. Це ім'я буде відповідати імені схеми, де створена збережена функція.

Типи об'єктів, для яких завжди потрібно вказати ім'я схеми:

- Таблиця;
- Перегляд;
- Функція;
- Послідовність;

MS SQL

```
SELECT *

  FROM [GOLD_TEST_SS].[dbo].[Account];
```

PostgreSQL

```
SELECT *

  FROM GOLD_TEST_SS_dbo.Account;
```

MS SQL

```
SELECT * FROM ..Bank;

SELECT * FROM .Bank;

IF 1 != 1
BEGIN
  INSERT INTO ..Bank(id, [Description], BIC)
  VALUES(100500, 'Stopitsot', 'STO500');

  INSERT INTO .Bank(id, [Description], BIC)
  VALUES(100501, 'StopitsotOdin', 'STO501');

  UPDATE ..Bank
  SET [Description] = CONCAT('***', [Description], '***')
  WHERE id = 100500;

  UPDATE .Bank
  SET [Description] = CONCAT('***', [Description], '***')
  WHERE id = 100501;

  DELETE ..Bank WHERE id = 100500;
  DELETE .Bank WHERE id = 100501;
END;
```

PostgreSQL

```
SELECT * FROM gold_test_ss_dbo.Bank;

SELECT * FROM gold_test_ss_dbo.Bank;

IF 1 != 1

THEN

  insert into gold_test_ss_dbo.Bank(id, "Description", BIC)

  values(100500, 'Stopitsot', 'STO500');


  insert into gold_test_ss_dbo.Bank(id, "Description", BIC)

  values(100501, 'StopitsotOdin', 'STO501');


  update gold_test_ss_dbo.Bank

  set "Description" = concat('***', "Description", '***')

  where id = 100500;


  update gold_test_ss_dbo.Bank
```

Рисунок 2.4 – Приклад міграції

2.2.3 Локальні змінні та параметри

Імена змінних MSSQL повинні починатися зі знаку (@). У PostgreSQL ім'я локальної змінної не може містити символ "@". Його слід замінити префіксом "var_".

MS SQL

```
DECLARE @testVariable INT;
```

PostgreSQL

```
DECLARE var_testVariable INT;
```

Рисунок 2.5 – Міграція змінних

Процедури в MSSQL повинні починатися з одного символу @, як у стандартній змінній Transact–SQL. У PostgreSQL ім'я параметра не може містити символ "@". Його слід замінити префіксом "par_".

MS SQL

```
CREATE PROCEDURE PrmExample
    @LastName nvarchar(50),
    @FirstName nvarchar(50)
AS
…
```

PostgreSQL

```
CREATE PROCEDURE PrmExample (
    par_LastName varchar(50),
    par_FirstName varchar(50)
)
BEGIN
…
```

Рисунок 2.6 – Міграція параметрів

2.2.4 Зарезервовані слова

Microsoft SQL Server використовує зарезервовані ключові слова для визначення, маніпулювання та доступу до баз даних. Зарезервовані ключові слова – це частина граматики мови Transact–SQL, яка використовується SQL Server для розбору та розуміння операторів та пакетів Transact–SQL. Хоча синтаксично можливо використовувати зарезервовані ключові слова SQL Server як ідентифікатори та імена об’єктів у сценаріях Transact–SQL, ви можете це зробити лише за допомогою обмежених ідентифікаторів.

Зарезервовані ключові слова Transact–SQL можуть використовуватися як ідентифікатори або імена баз даних або об'єктів бази даних, таких як таблиці, стовпці, представлення даних тощо. Використовуйте або цитовані

ідентифікатори, або ідентифіковані ідентифікатори. Використання зарезервованих ключових слів як імен змінних та збережених параметрів процедури не обмежується.

У PostgreSQL є кілька слів, які зарезервовані у своєму синтаксисі. Якщо є деякі назви об'єктів, які відповідають цим зарезервованим словам, вам потрібно обернути їх символами "". Але не так просто. Є кілька правил, як. PostgreSQL дозволяє імена, що складаються з одного ідентифікатора або декількох ідентифікаторів. Компоненти багаточастинного імені повинні бути розділені символами періоду (".") У PostgreSQL ви можете звернутися до стовпця таблиці, використовуючи будь–яку з наведених нижче форм.

Column Reference	Meaning
col_name	The column col_name from whichever table used in the statement contains a column of that name.
tbl_name.col_name	The column col_name from table tbl_name of the default database.
db_name.tbl_name.col_name	The column col_name from table tbl_name of the database db_name.

Рисунок 2.7 – Зарезервовані слова PostgreSQL

2.2.5 Приклад перетворення для операцій з типами VARBINARY

Бінарні константи в MSSQL мають префікс 0x і є рядком шістнадцяткових чисел. Вони не вкладаються в лапки. Нижче наведено приклади двійкових рядків:

```
0xAE
0x12Ef
0x69048AEFDD010E
0x   (empty binary string)
```

Формат «escape» – це традиційний формат PostgreSQL для типу bytea. Для цього потрібен підхід представлення двійкової рядка як послідовності символів ASCII, при цьому перетворює ті байти, які неможливо представити як символ ASCII, у спеціальні послідовності відходу.

При введенні значень байтів у формат евакуації необхідно оминути певні значення октетів, тоді як усі октетні значення можна уникнути. Загалом, щоб уникнути октету, перетворіть його у його трицифрове вісімкове значення та передуйте його зворотній косої риси (або двох зворотних косих рисочків, якщо записувати значення як буквальне, використовуючи синтаксис рядового рядка). Сам зворотний проріз (значення октету 92) може бути представлений подвійними нахилами.

PostgreSQL використовує Hex Format. Формат "шістнадцятковий" кодує двійкові дані у вигляді 2–х шістнадцяткових цифр на один байт, найзначніший спочатку. Цілому рядку передує послідовність \ x (щоб відрізнити його від формату). У деяких контекстах може виникнути необхідність уникнути початкового косого кута, подвоївши його, у тих самих випадках, коли косої косої риси потрібно подвоїти у форматі евакуації; деталі відображаються нижче. Шістнадцяткові цифри можуть бути як верхніми, так і малими, а пробіл дозволений між парами цифр (але не в межах пари цифр і не в початковій \ x послідовності). Шестнадцятковий формат сумісний із широким колом зовнішніх програм та протоколів, і це, як правило, швидше конвертувати, ніж формат виходу, тому його використання є кращим.

MS SQL	PostgreSQL
0xAE	E'\\xAE'
0x12Ef	E'\\x12Ef'
0x69048AEFDD010E	E'\\x69048AEFDD010E'
0x00	

Рисунок 2.8 – Приклад міграції даних

2.3 Порівняння схем баз даних

Для розроблювального підходу до міграції даних порівняння схем баз даних є допоміжним етапом, результат якого буде підказкою для людини при завданні правил переносу. Результатом порівняння будуть оцінки подібності відносин з однієї схеми з кожним відношенням з іншої схеми – міри схожості.

Пропонований метод порівняння схем баз даних грунтується на обчисленні заходів подібності згідно послідовностям обходу схем БД і використовує такі відомості про відношення:

- імена відношень;
- імена атрибутів;
- тип атрибута;
- довжина атрибута;
- значення за замовчуванням;
- автоінкрементні значення;
- можливий, первинний, зовнішні ключі;
- допустимість невизначених значень;
- тригери;
- обмеження цілісності типу «CHECK»;
- інші.

Для обчислення міри схожості двох відношень X і Y пропонується використовувати наступну формулу:

$$\phi(X, Y) = c_{name} \times \phi_{name}(X, Y) + c_{attributes} \times \phi_{attributes}(X, Y) + c_{pkuks} \times \phi_{pkuks}(X, Y) + c_{fks} \times \phi_{fks}(X, Y) + c_{sqls} \times \phi_{sqls}(X,Y) \quad (2.1)$$

де cname, cattributes, cpkuks, cfks, csqls – вагові коефіцієнти, значення яких невід'ємні і їх сума дорівнює 1, а φname (X, Y), φattributes (X, Y), φpkuks (X, Y), φfks (X, Y), φsqls (X, Y) – функції обчислення заходів подібності імен, атрибутів, первинного та можливого ключів, зовнішніх ключів, тригерів і обмежень цілісності типу «CHECK», значення яких лежать в інтервалі [0, 1]. Таким чином, значення міри схожості двох відносин лежить в інтервалі від [0, 1]. Вагові коефіцієнти можуть бути обрані відповідно до рівноважного критерію, статистичними даними або задані користувачем.

Для обчислення міри подібності імен відношень і атрибутів можуть бути використані різні способи: порівняння на основі максимальної підрядка, відстані Левенштейна, використання словників синонімів, абревіатур і інші. Кінцева міра подібності імен вибирається як максимальне значення серед значень, отриманих вибраними способами.

Заходи подібності атрибутів, можливих і первинного, зовнішніх ключів можуть бути обчислені за наступними формулами, в яких відповідні вагові коефіцієнти сі невід'ємні і їх сума дорівнює 1.

Порівняння тригерів і обмежень цілісності типу «CHECK» вимагає додаткового теоретичного дослідження, але можливе використання такої формули для обчислення заходів подібності зазначених обмежень цілісності: Порівняння схем баз даних (S1 і S2) відбувається в два етапи. На першому етапі будуються послідовності обходу порівнюваних схем. На другому етапі на кожному кроці вибирається нерозглянутих відношення зі схеми S1 згідно послідовності її обходу і обчислюються міри схожості з кожним відношенням зі схеми S2 в порядку, визначеному послідовністю обходу схеми S2. Необхідність обчислення заходів

подібності відносин згідно послідовностям обходу схем обумовлюється тим, що міра подібності зовнішніх ключів залежить від міри схожості батьківських відношень. Якщо в схемах є циклічні залежності за зовнішніми ключам, то при обчисленні заходів подібності зовнішніх ключів відповідних відношень використовується наближене значення міри схожості батьківських відносин.

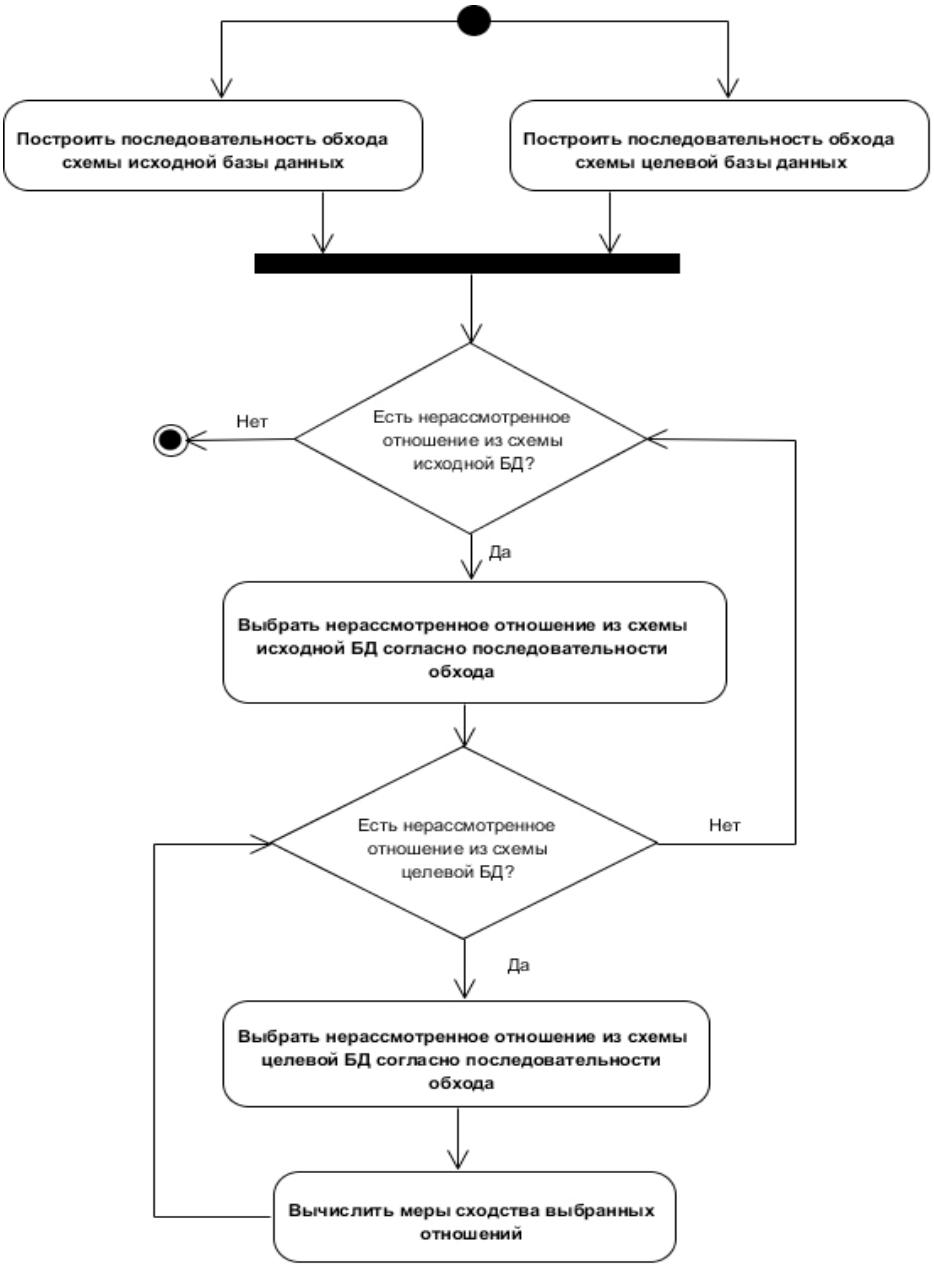


Рисунок 2.9 – Діаграма діяльності порівняння схем баз даних

Діаграма діяльності порівняння схем баз даних послідовності обходу схем вихідної і цільової баз даних будуються на одному з початкових етапів процесу

міграції в зв'язку з їх використанням не тільки в порівнянні схем, але і в механізмі перенесення даних.

3 ДОДАТОК ДЛЯ МІГРАЦІЇ ДАНИХ

3.1 Технології для реалізації програми

Для реалізації програми для міграції даних було обрано такі технології:

- технологія Java [33];
- база даних HSQLDB [16];
- брокер Hibernate [14];
- платформа Eclipse RCP [28, 43].

Технологія Java – це і мова програмування, і платформа. Мова програмування Java є потужним високорівневим і універсальною мовою, які базуються на парадигмі об'єктно–орієнтованого програмування. Платформа

Java складається з двох компонентів: Java Virtual Machine (JVM) і Java Application Programming Interface (API). Вихідний код програми на Java транслюється в байт–код, що виконується на JVM. JVM являє собою програму, реалізовану для різного апаратного забезпечення і багатьох операційних систем, що забезпечує кроссплатформенность програм на Java. JVM поставляється з набором стандартних бібліотек, в сукупності з якими становить Java Runtime Environment (JRE). Недоліком такого способу виконання програм часто називають зниження продуктивності через інтерпретації байт–коду віртуальною машиною. Однак ряд удосконалень, реалізованих в більш пізніх версіях Java, дозволяє досягти продуктивності, близькою до продуктивності «native» коду, не відбиваючись негативно на такій характеристиці як переносимість. На сьогоднішній день Java включає багато різних служб, технологій, компонентів, наприклад, JDBC, JMF, RMI, JINI, JNI, що також полегшує написання програм.

Для зберігання словників, використовуваних в порівнянні схем баз даних,

була обрана база даних HSQLDB (HyperSQL DataBase) – реляційна база даних, написана на Java з відкритим вихідним кодом. Вона поширюється по власної ліцензії, близькою до ліцензії BSD, і сумісна з усіма основними ліцензіями програмного забезпечення з відкритим вихідним кодом. HSQLDB підтримує майже повністю стандарт ANSI–92 SQL і основні положення стандарту SQL: 2008. Вона відрізняється невеликим розміром і надає швидкий, багато–і підтримує транзакції процесор бази даних. HSQLDB може використовуватися і як окремий сервер з підтримкою мережевих з'єднань по JDBC, і як вбудована СУБД.

Для забезпечення роботи зі словниками обраний Hibernate – високо продуктивний брокер, призначений для вирішення завдань об'єктно–реляційного відображення, для Java. Hibernate є вільним програмним забезпеченням з відкритим вихідним кодом, поширюваним на умовах GNU LGPL, і надає легкий у використанні фреймворк, найбільш корисний при використанні об'єктно–орієнтованих моделей предметних областей і бізнес логіки в проміжному шарі, написаному на Java.

Eclipse RCP (Eclipse Rich Client Platform) – платформа, що забезпечує можливість створення додатків з використанням потужної універсальної оболонки при мінімальній кількості необхідних плагінів. Ідея цієї технології виникла при розробці чергової версії Eclipse IDE і з'явилася у версії 3. Eclipse IDE – інтегроване середовище розробки, що підтримує широкий спектр інструментаріїв для мов і систем і володіє такими характеристиками, як кроссплатформенность, модульність і іншими. До версії 3 в Eclipse IDE підтримувалася розробка плагінів тільки для розширення безпосередньо середовища розробки Eclipse, а з версії 3.0 Eclipse перестав бути монолітною IDE, що підтримує розширення, а сам став набором розширень. Використання фреймворків OSGi і SWT / JFace, що лежать в основі, дозволило створити платформу для розробки повноцінних клієнтських додатків RCP. Така платформа являє собою мінімальний набір плагінів, необхідних для створення RCP додатки і використовуються разом. Поверх цієї платформи була створена платформа Eclipse, що представляє собою набір розширень RCP – перспективи, редактори,

навігатори, модуль Java Development Tools (JDT) та інші. Отримана платформа має ряд переваг, що обумовлюють її придатність для розробки повноцінних клієнтських додатків: модульність, кроссплатформенность, підтримка багатомовності, безкоштовність, велика кількість існуючих полігонів, бібліотек і фреймворків. Ліцензія Eclipse Public License дозволяє використовувати створені додатки в комерційних цілях. Отримана платформа має ряд переваг, що обумовлюють її придатність для розробки повноцінних клієнтських додатків: модульність, кроссплатформенность, підтримка багатомовності, безкоштовність, велика кількість існуючих полігонів, бібліотек і фреймворків. Ліцензія Eclipse Public License дозволяє використовувати створені додатки в комерційних цілях. Отримана платформа має ряд переваг, що обумовлюють її придатність для розробки повноцінних клієнтських додатків: модульність, кроссплатформенность, підтримка багатомовності, безкоштовність, велика кількість існуючих полігонів, бібліотек і фреймворків. Ліцензія Eclipse Public License дозволяє використовувати створені додатки в комерційних цілях.

3.2 Особливості реалізації порівняння схем баз даних

На реалізацію запропонованого методу для порівняння схем баз даних в додатку, що розробляється вплинули обрані технології. Мова Java зумовив використання JDBC [20] в реалізації програми. JDBC дозволяє використовувати вбудоване відображення типів даних різних СУБД в типи даних JDBC, а також отримувати додаткові відомості про відношення, атрибутах, типах даних: автоінкрементірованіе значення (autoincrement), можливість використання атрибута з певним типом даних в реченні SQL WHERE (searchable), допустимість невизначених значень (nullable), кількість дрібних знаків (decimalDigits) якщо для

типу даних це може бути застосовано, чутливість до регістру символів (caseSensitive), загальна кількість значущих цифр (precision), можливість використання тип а даних для грошових значень (fixedPreScale).

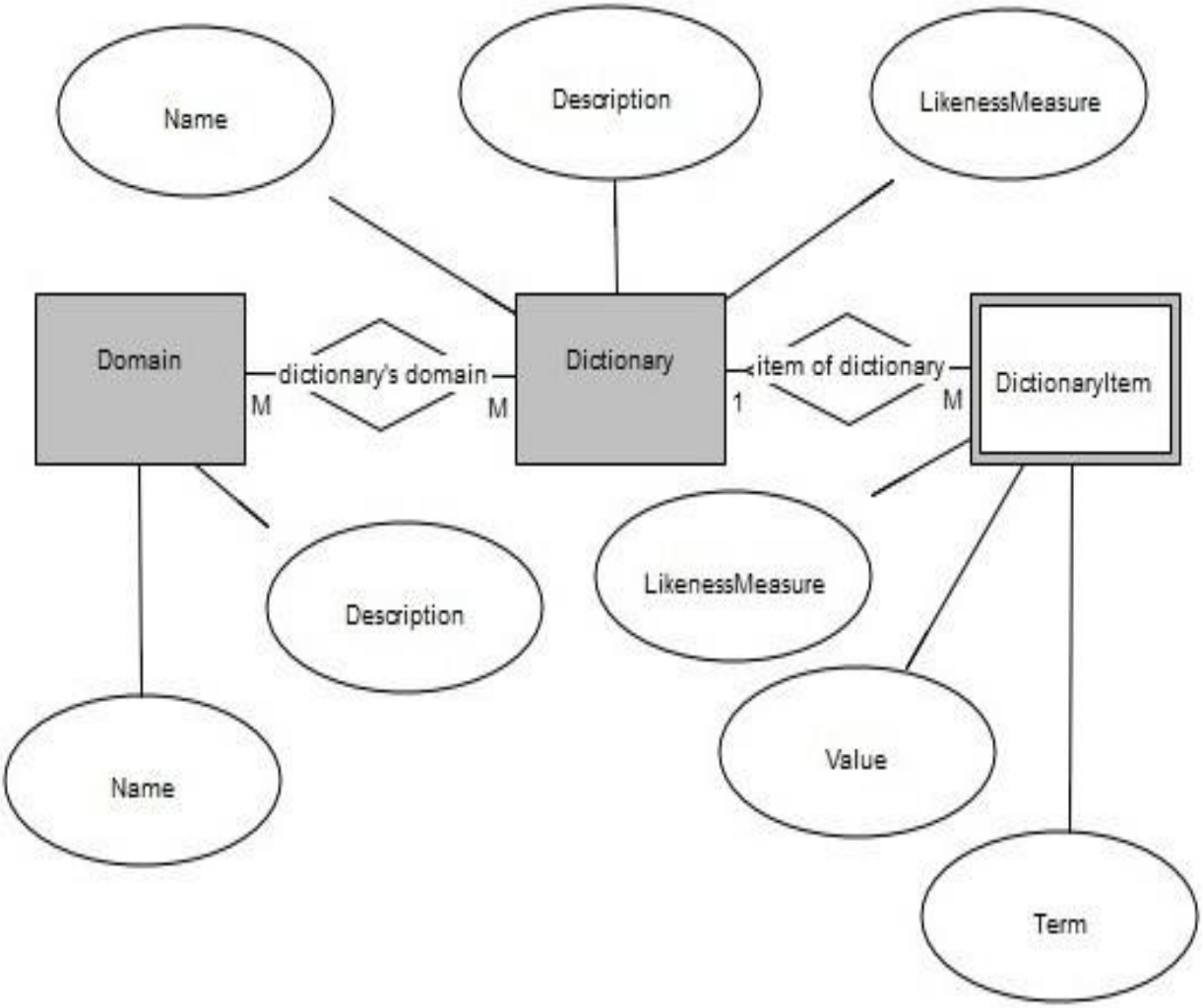


Рисунок 3.1 – ER–діаграма словників для порівняння імен відношень і атрибутів

Для обчислення міри подібності імен відношень і атрибутів були реалізовані наступні способи: порівняння на основі максимальної підрядка, відстані Левенштейна [48], використання словників (відповідна ER–діаграма яких зображена на рисунку 3.1), порівняння на рівність з урахуванням / без урахування регістру. Діаграма класів для порівняння імен відношенвтер і атрибутів наведена на рисунку 3.2.

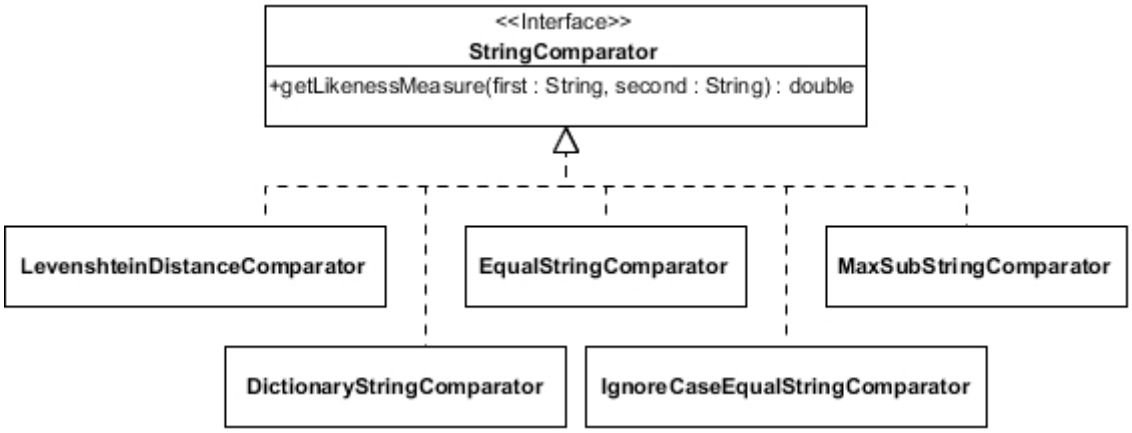


Рисунок 3.2 – Діаграма класів для порівняння імен відношень і атрибутів

Для обробки додаткової інформації про імена відношень і атрибутів, що дозволяє поліпшити результат порівняння, наприклад, про додавання префіксів, можуть бути використані перетворювачі, що здійснюють попередню обробку порівнюваних рядків. Реалізовані на даний момент перетворювачі показані на діаграмі класів, наведеної на рисунку 3.3.

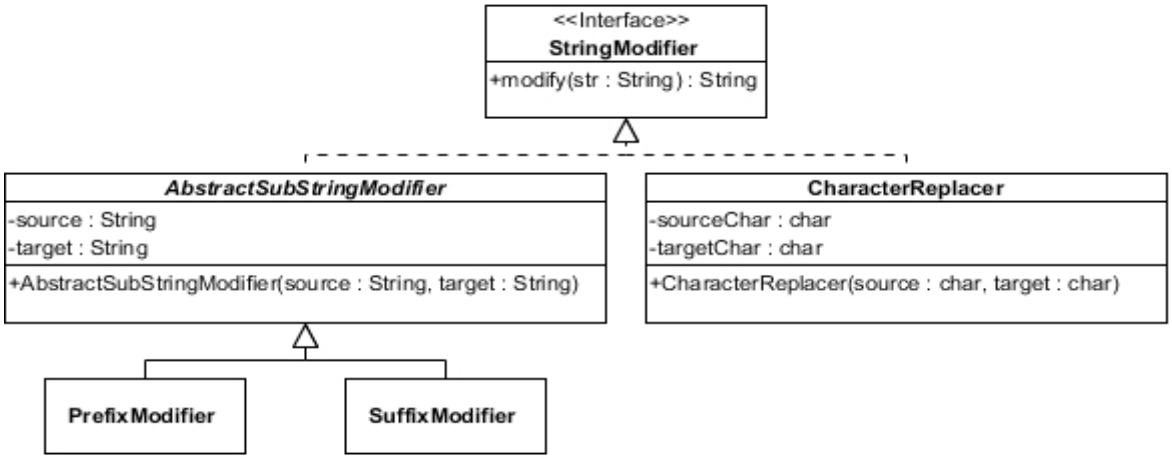
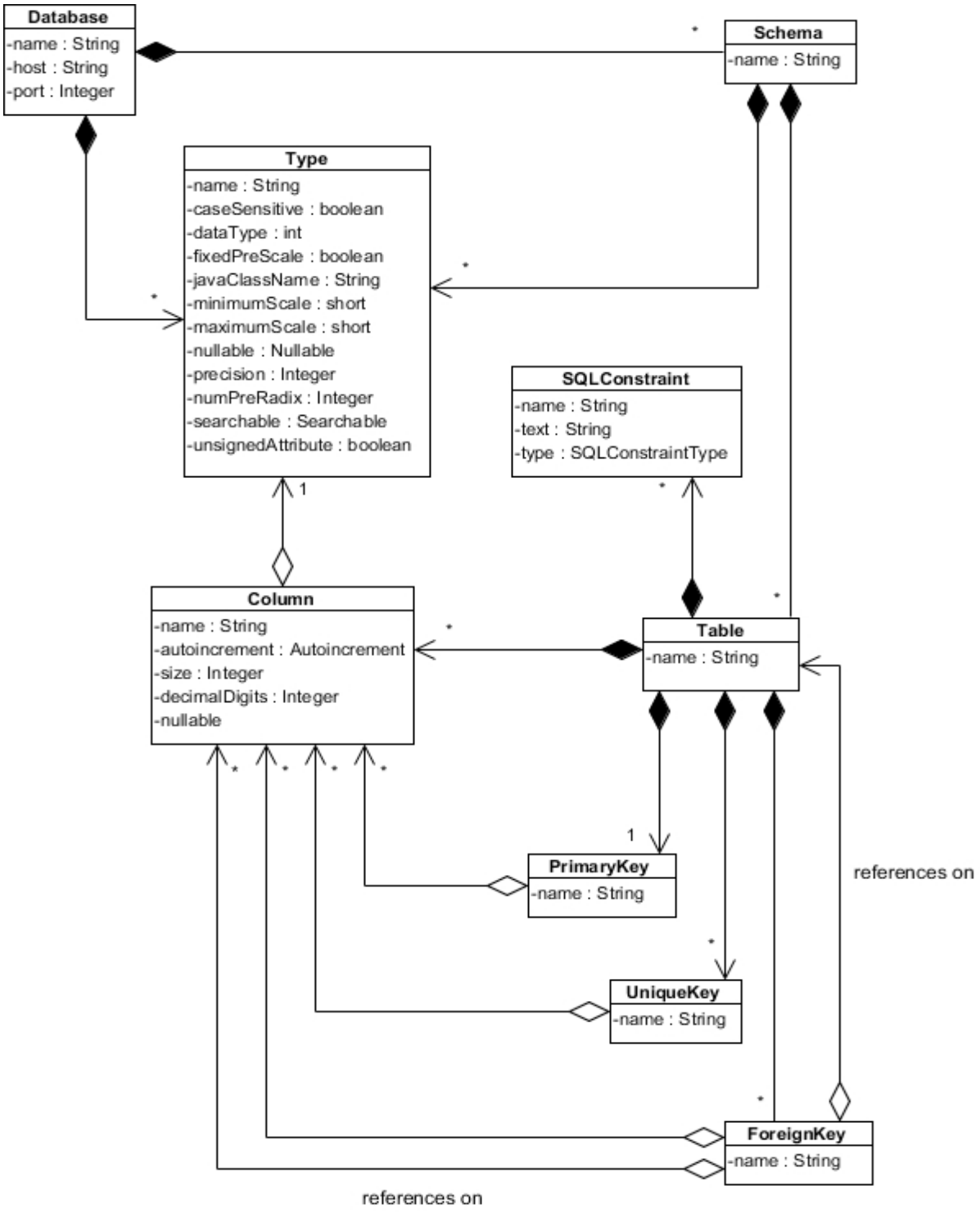


Рисунок 3.3 – Діаграма класів для попередньої обробки імен відношень та атрибутів

Для подання схем баз даних в оперативній пам'яті у вигляді, що містить необхідні відомості і приховує деталі роботи з конкретними СУБД, використовується модель, діаграма класів якої приведена на рисунку 3.4. На даній діаграмі для зручності відображення опущені операції класів. Клас Schema на діаграмі відповідає поняттю «схема», використовуваному в деяких СУБД (наприклад, PostgreSQL [40]).



Обчислення заходів подібності відношень забезпечується класами і інтерфейсами, зображеними на рисунку 3.5.

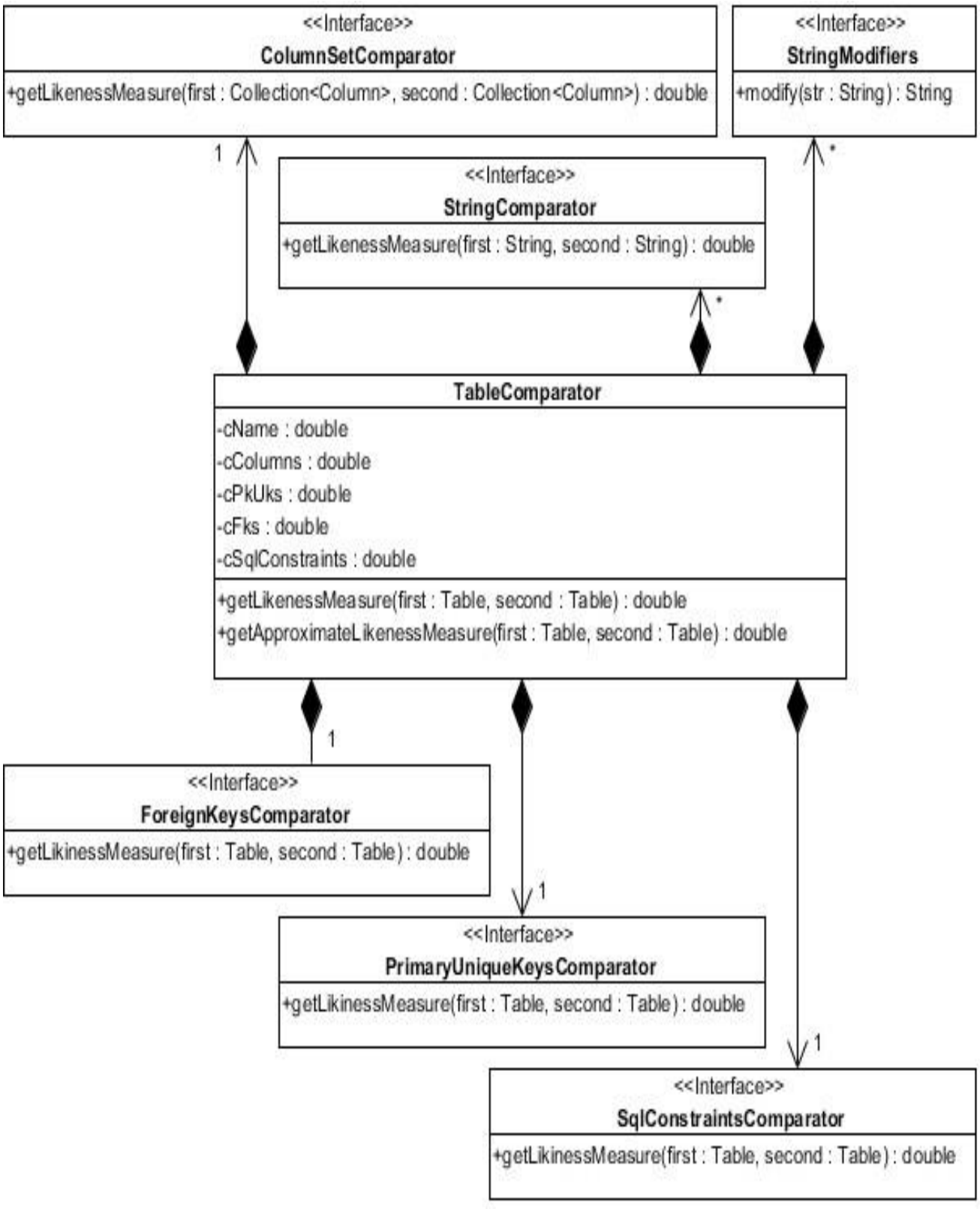


Рисунок 3.5 – Діаграма класів для обчислення заходів подібності

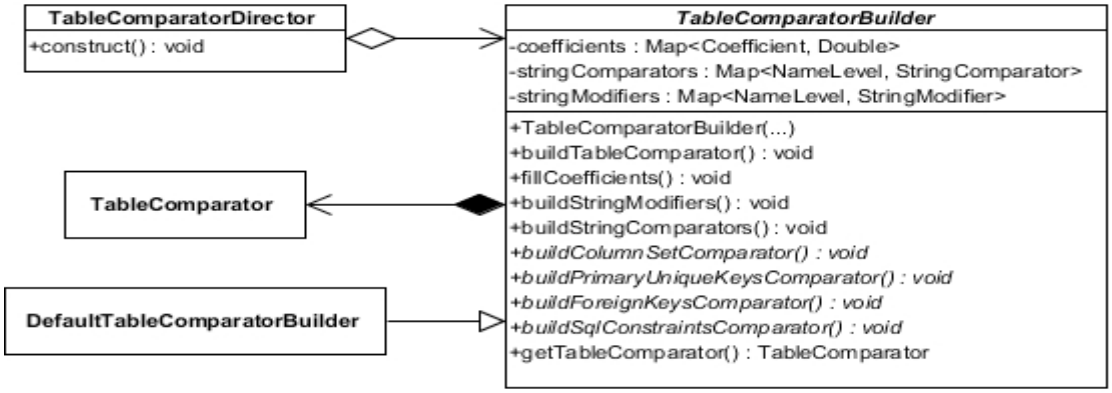


Рисунок 3.6 – Діаграма класів для обчислення заходів подібності

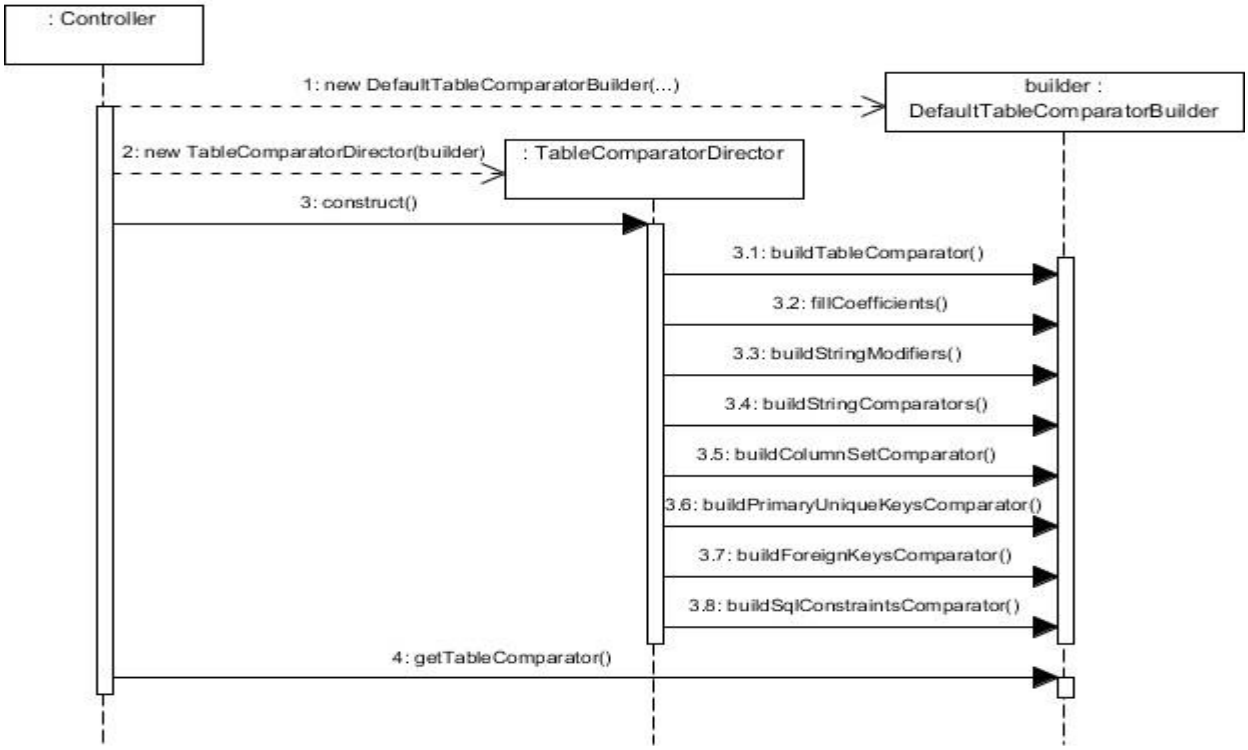


Рисунок 3.7 – Діаграма класів для обчислення заходів подібності

3.3 Демонстрація міграції за допомогою додатку

Додаток для міграції даних можна використовувати без доступу до інтернету, лише завантаживши JVM до ПК та відкрити jar.файл. Інтерфейс програми зроблений приємним і зрозумілим для користувача програми.

Рисунок 3.8 – Головна сторінка програми

Користувач може завантажити свій xml файл або відкрити вже існуючий проект для виконання міграції. У вкладці налаштування можна обрати кудли буде приходити файл логів при виконанні міграції

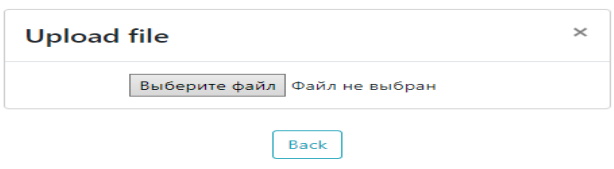


Рисунок 3.9 – Вкладка “Завантажити файл”

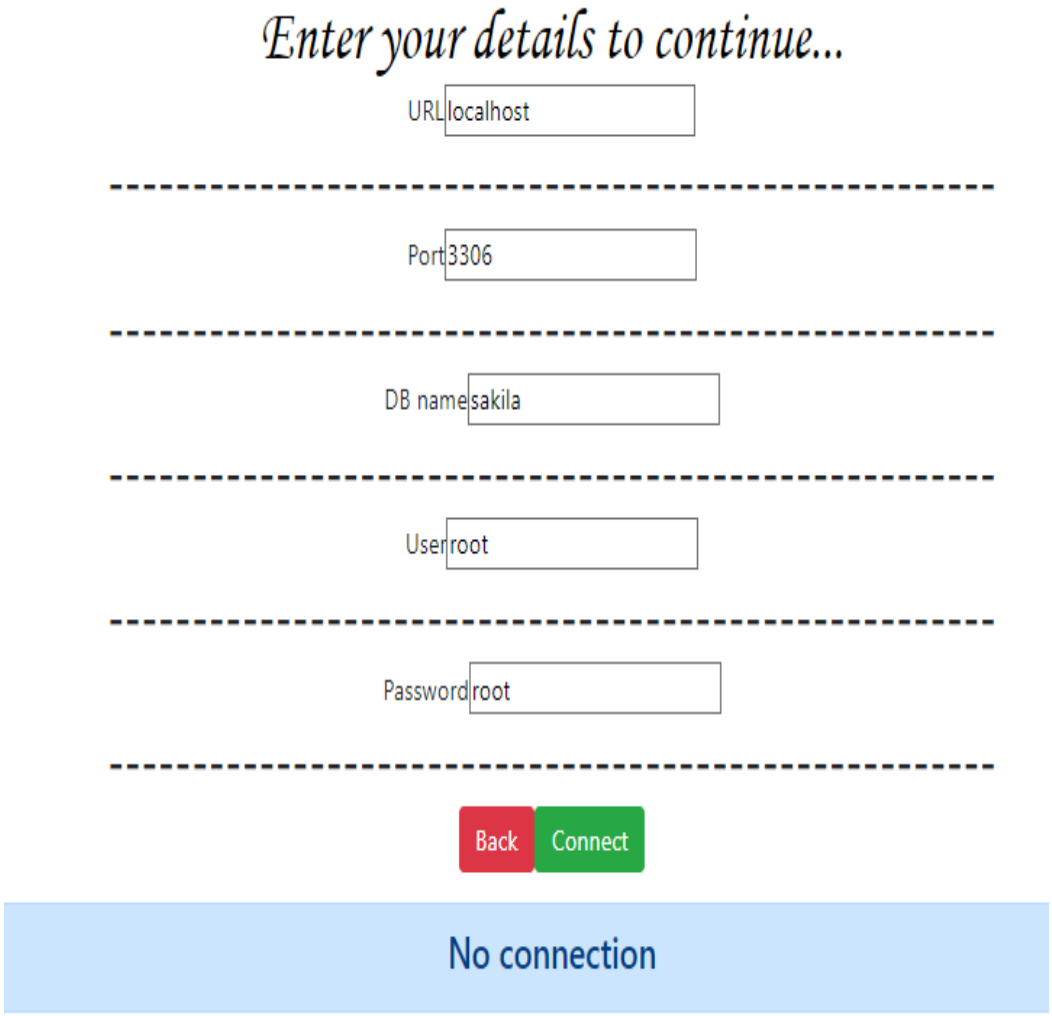


Рисунок 3.10 – Вкладка “Деталі БД”

У результати завантаження схеми бази даних будується дерево бази даних і при натисканні на існуючий об’єкт будується таке ж саме дерево у вихідній базі даних. У центральному вікні завантажується перетворений скрипт.

Бази даних представлені у вигляді структури даних дерево, кожна база даних зеркально відображає схему даних. При натісканні на об’єкт бази даних у центральному вікні відображається скрипти на базу даних PostgreSQL та MS Server.

Рисунок 3.11 – Результат роботи програми

У результаті виконання програми дані з бази MS Server були повністю мігріровані до бази PostgresSQL без втрати даних.

ВИСНОВКИ

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

- розширений механізм перенесення даних для виконання паралельної міграції даних;
- розроблений і реалізований алгоритм побудови послідовності обходу схеми бази даних;
- розроблений і реалізований алгоритм порівняння схем баз даних;
- створений виконуваний еволюційний прототип додатка для міграції даних.

У результаті виконання атестаційної роботи було проведено дослідження міграції даних з MS Server до PostgreSQL.

У рамках атестаційної магістерської роботи була доведена актуальність дослідження, проаналізовані існуючі аналоги системи, обґрунтована мета розробки додатку для міграції, який розглядається в цієї роботі та сформовані критерії ефективності, поставлена задача дослідження.

На основі цих досліджень був розроблений алгоритм міграції данних, виконаний процесс міграції з MS Server до PostgeSQL.

Задля демонстрації міграції та його практичного використання було розроблене демонстраційне автоматизоване програмне забезпечення.

У майбутньому планується продовження досліджень на тему міграції даних, а саме розширення мов баз даних. За результатами дослідження можна зробити висновок що поставлена задача була успішно виконана, міграція даних була успішно виконана та проведені досліди.

ПЕРЕЛІК ПОСИЛАНЬ

1. 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.

2. Algergawy A. Understanding the schema matching problem / A. Algergawy, E. Schallehn, G. Saake // Proceedings of 7th Conference on 7th WSEAS International Conference on Applied Computer Science (ACS'07). Venice, Italy, November 21–23, 2007. – Stevens Point, Wisconsin, USA, 2007. – Vol. 7. – P. 59–68.

3. Anan M. Managing uncertainty in schema matcher ensembles / M. Anan, A. Gal // Scalable Uncertainty Management First International Conference (SUM 2007). Washington, DC, USA, October 10–12, 2007. Proceedings. – Springer Berlin / Heidelberg, 2007. – P. 60–73.

4. 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).

5. Batini C. A comparative analysis of methodologies for database schema integration / C. Batini, M. Lenzerini, S. Navathe // ACM Computing Surveys. – 1986. – Vol. 18, Issue 4. – P. 323– 364.

6. 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.

7. Bunke H. Graph matching: theoretical foundations, algorithms and applications // Proceedings of Vision Interface. Montreal, Quebec, Canada, May 14–17, 2000. – 2000. – P. 82–88.

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

9. Data migration [Electronic resource] // Data Integration Info. – Electronic data. – URL:http://www.dataintegration.info/data–migration (Reference date: 01.12.2011).

10. Data migration best practices [Electronic resource] // NetApp. – URL:http://partners.netapp.com/go/techontap/NGS_migration.pdf (Reference date: 10.01.2011).

11. 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.

12. Fishman D. Database migration: the key to unlocking your business assets [Electronic resource] // Narayana Vyas Kondreddi's website. – URL:http://vyaskn.tripod.com/database_migration_white_paper.htm (Reference date: 28.09.2010).

13. 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.

14. Hibernate – JBoss Community [Electronic resource] – Electronic data. – URL:http://www.hibernate.org/ (Reference date: 04.02.2012).

15. Howard P. Data migration in the global 2000 – research, forecasts and survey results – a survey paper by Bloor Research (2007) [Electronic resource] / P. Howard, C. Potter // Rever.eu. – Electronic data. – URL:http://www.rever.eu/white–papers/876–Data–Migration–Survey.pdf (Reference date: 15.04.2012).

16. HSQLDB [Electronic resource]. – Electronic data. – URL:http://hsqldb.org/ (Reference date: 04.02.2012).

17. Hudicka J. An overview of data migration methodology [Electronic resource] //Dulcian, Inc.–Electronicdata.
http://www.dulcian.com/Articles/Overview_Data_Migration_Methodology.
(reference URL:date:05.09.2010).

18. Hudicka J. DataMIG – a data migration management tool suite [Electronic resource]. – URL:http://www.dulcian.com/Articles/DataMIG.htm (Reference date: 05.09.2010).

19. Hudicka J. So, you say your data's clean? [Electronic resource] // Dulcian
URL:http://www.dulcian.com/papers/IOUG/1999/SoYouSayDatasClean.htm
(Reference date: 28.09.2010).

20. Java SE 7 Java Database Connectivity (JDBC) – related APIs & Developer Guides//OracleDocumentation.–Electronicdata.–URL: http://docs.oracle.com/javase/7/docs/technotes/guides/jdbc/index.html (Reference date: 28.02.2012).

21. Kashyap V. Semantic and schematic similarities between database objects: a context– based approach / V. Kashyap, A. Sheth // The VLDB Journal. – 1996. – Vol. 5, Issue 4. – P. 276–304.

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

23. Liquibase | Database Refactoring [Electronic resource] – Electronic data. – URL:http://www.liquibase.org/ (Reference date: 05.04.2011).

24. 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.

25. 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.

26. Стаття з вікіпедії. // https://uk.wikipedia.org/wiki/Сховище_даних