GraphMigr8

Software Requirements Specification

GraphMigr8 1.0

25.05.2025

Mykhnevych Dmytro

Software Engineer

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Description** | **Author** | **Comments** |
| 25.05.2025 | GraphMigr8 initial release | Mykhnevych D. | Version 1.0. |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Document Approval**

The following Software Requirements Specification has been accepted and approved by the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **Signature** | **Printed Name** | **Title** | **Date** |
|  | Mykhnevych D. | Software Engineer | 26.05.2025 |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

**REVISION HISTORY II**

**DOCUMENT APPROVAL II**

**1. INTRODUCTION 1**

1.1 Purpose 1

1.2 Scope 1

1.3 Definitions, Acronyms, and Abbreviations 2

1.4 Overview 3

**2. GENERAL DESCRIPTION 4**

2.1 Product Perspective 4

2.2 Product Functions 6

2.3 User Characteristics 6

2.4 General Constraints 7

2.5 Assumptions and Dependencies 8

**3. SPECIFIC REQUIREMENTS 9**

3.1 External Interface Requirements 9

*3.1.1 User Interfaces* 9

*3.1.2 Hardware Interfaces* 9

*3.1.3 Software Interfaces* 9

*3.1.4 Communications Interfaces* 9

3.2 Functional Requirements 9

*3.2.1 Імпорт реляційних даних* 9

*3.2.2 Міграція даних у графову модель* 10

3.3 Use Cases 11

*3.3.1 Взаємодія користувачів з системою* 11

3.4 Classes / Objects 11

*3.4.1 Таблиця 11*

*3.4.2 Запит 1*2

3.5 Non-Functional Requirements 12

*3.5.1 Performance* 12

*3.5.2 Reliability* 12

*3.5.3 Availability 12*

*3.5.4 Security 1*3

*3.5.5 Maintainability 1*3

*3.5.6 Portability* 13

3.6 Inverse Requirements 13

3.7 Design Constraints 13

3.8 Logical Database Requirements 14

**4. ANALYSIS MODELS** 15

4.1 Sequence Diagrams 15

**5. CHANGE MANAGEMENT PROCESS** 16

**1. Introduction**

**1.1 Purpose**

Метою цього документа є забезпечення повного та чіткого представлення продукту GraphMigr8 та подання всіх його функціональних та нефункціональних вимог. Документ призначений для команд, які реалізовуватимуть та перевірятимуть продукт GraphMigr8. SRS складений для забезпечення однакового розуміння як розробниками, так і бізнес-аналітиками цілей розробки даного продукту та його завдань. У процесі розробки GraphMigr8 документ оновлюватиметься, таким чином можна буде стежити за всіма етапами розробки продукту.

GraphMigr8 – це інструмент, який дозволяє здійснити міграцію даних з реляційної СУБД MSSQL в графову СУБД Neo4J за допомогою різних методів міграції. Також можливо переглянути основні метрики, які були зібрані під час міграції тим чи іншим методом (час міграції, обсяг використаної оперативної пам’яті, використання CPU).

**1.2 Scope**

GraphMigr8 забезпечує підтримку ефективного перенесення інформації між традиційною реляційною СУБД Microsoft SQL Server та сучасною графовою базою даних Neo4j. Проєкт охоплює аналіз існуючих методів міграції, розробку нових оптимізованих підходів, експериментальне дослідження ефективності міграції, а також створення програмного забезпечення для реалізації процесу міграції.

Мета цього проєкту полягає у проведенні дослідження існуючих підходів до міграції даних з реляційної в графову базу даних з подальшим удосконаленням розглянутих методів.

**1.3 Definitions, Acronyms, and Abbreviations**

|  |  |
| --- | --- |
| Термін | Визначення |
| Веб-платформа | Багатоцільова платформа для розробки веб-проектів та управління вмістом |
| Алгоритм міграції | Набір правил і операцій, що забезпечують коректне перетворення реляційних структур у графову модель даних |
| Вершина | Основний елемент графової моделі даних, що представляє сутність з набором атрибутів |
| Ребро | Елемент графової моделі, що визначає зв’язки між вершинами і може містити власні атрибути |
| Сутність | Об'єкт даних у реляційній моделі, що представлений у вигляді таблиці |
| Front-end/ Клієнтський додаток | Графічний інтерфейс, який відображається користувачу за допомогою фреймворку Angular (для веб-версії), який забезпечує зв’язок користувача та можливість отримати інформацію з серверу |
| Back-end/Серверна частина | Додаток, який взаємодіє з базою даних і виконує основні обчислення та маніпуляції з даними. Зв’язок між клієнтами та сервером здійснюється через протокол HTTPS |
| API | Інтерфейс або протокол зв’язку між клієнтом і сервером, призначений для полегшення створення програмного забезпечення на стороні клієнта |

**1.5 Overview**

SRS ділиться на 5 логічних частин: вступ, загальний опис, конкретні вимоги, аналіз моделей та організація внесення змін.

У другому розділі представлено огляд функціональних можливостей системи та її взаємодії з іншими системами. У цьому розділі також описано функції, доступні користувачеві, та обмеження його можливостей. Цей розділ також містить обмеження системи та припущення щодо майбутнього продукту.

У третьому розділі детально представлено специфікацію вимог і опис різних інтерфейсів системи.

Глава четверта присвячена пріоритетності вимог.

Розділ п'ятий описує, як змінюються та затверджуються вимоги до проекту. Він містить загальну інформацію про процес управління.

**2. General Description**

Продукт представляє собою програмне забезпечення для автоматизації процесу міграції даних з реляційної в графову базу даних. Його основна функція – забезпечити коректне перетворення структурованої інформації, що зберігається у Microsoft SQL Server, у формат, який відповідає графовій моделі даних, зокрема Neo4j.

**2.1 Product Perspective**

Було проведено аналіз ринку, який показав, що існує низка методів для роботи з графовими базами даних та міграції даних з реляційних структур у графову модель.

Серед методів міграції реляційних даних в графову модель доволі популярними є методи Roberto De Virgilio, Yelda Ünal та інші. Окремо варто виділити групу методів, які спираються на ER-модель БД та по своїй суті є схожими на методи логічного проєктування графових БД на основі ER-моделі. Яскравим представником такого напряму є метод Чжихонг Нань та Сюе Бай. Між тим, не варто вважати такі методи повноцінними представниками методів міграції між реляційною та графовою БД.

Доволі новим та перспективним методом міграції реляційної БД в графову є метод Rel2Graph. Метод Rel2Graph заснований на класифікації таблиць реляційної БД на таблиці сутностей та таблиці зв’язування, та їх відповідного перетворення на структури графової моделі. Так, таблиця вважається таблицею сутностей, якщо взагалі не має зовнішніх ключів, або є один чи більше ніж два зовнішніх ключі, або наявні два зовнішні ключі з простим первинним ключем. Таблиця зв'язування визначається наявністю двох зовнішніх ключів, первинний ключ якої або не є простим, або складається з цих двох зовнішніх ключів. Першим кроком алгоритм визначає, які таблиці є таблицями сутностей, а які таблиці використовуються для створення зв’язків (наприклад, проміжна таблиця в зв’язку багато до багатьох). Далі відбувається перетворення таблиць в графові структури. Так, кожна таблиця сутностей трансформується у вершини графа. Назва таблиці стає міткою вершини, а кожен її рядок перетворюється на окрему вершину. Атрибути таблиці при цьому стають атрибутами вершин. Таблиці зв'язування перетворюються на ребра графа: назва таблиці стає типом ребра, її рядки стають безпосередньо ребрами, а атрибути перетворюються на атрибути ребер. Додатково в графі створюються ребра типу HAS для відображення зв'язків один-до-одного та один-до-багатьох.

Аналіз методу показав певні недоліки, які можуть привести до неефективної міграції, а саме:

- таблиці, що мають два зовнішніх ключа та простий первинний ключ класифікуються як таблиці сутності, хоча дуже часто такі таблиці проєктуються саме для реалізації зв’язку багато до багатьох, тобто мають лише три стовбці – первинний ключ (частіше за все генерується автоматично) та два зовнішніх ключа (посилання на дві таблиці, що зв’язуються);

- таблиці, що мають два зовнішніх ключа та складовий первинний ключ класифікуються як таблиці зв’язування, хоча можливі таблиці сутностей, які мають два зовнішніх ключа та складовий первинний ключ.

Також алгоритм приймає на вхід файл з SQL-скриптами для створення реляційної БД, а не підключається до існуючої БД, витрачаючи таким чином час на обробку та парсінг текстового файлу.

**2.2 Product Functions**

Система дозволяє користувачу виконувати міграцію даних з реляційної бази даних Microsoft SQL Server у графову базу Neo4j, забезпечуючи коректне перетворення структурованої інформації. Вона надає можливість створення графової моделі на основі реляційних таблиць, підтримує автоматичне налаштування зв’язків між сутностями та забезпечує збереження семантичної цілісності даних.

Користувач може налаштовувати параметри міграції, обираючи алгоритм перетворення та модифікуючи правила відображення реляційних структур у графовий формат. Передбачено механізм верифікації коректності трансформації, що дозволяє перевіряти відповідність отриманих графових структур вихідним реляційним даним.

Система підтримує створення запитів для графової БД, що полегшує роботу з мігрованими даними, а також надає можливість аналізу ефективності проведеної міграції, забезпечуючи порівняння продуктивності різних методів та алгоритмів. Завдяки інтуїтивному інтерфейсу та адаптивності система дозволяє користувачам ефективно керувати процесом міграції, мінімізуючи часові витрати та ризик втрати даних.

**2.3 User Characteristics**

Користувачі розроблюваного продукту можуть бути розділені на кілька основних категорій залежно від їхнього досвіду роботи з базами даних, потреб у міграції та рівня технічної компетентності.

Перша категорія включає дослідників і науковців, які працюють з великими обсягами структурованих даних і потребують ефективного способу їх перетворення для проведення аналізу взаємозв’язків між сутностями. Вони зацікавлені у високій точності міграції, збереженні семантичної цілісності та можливості виконання складних запитів у графовій моделі.

Друга категорія представлена розробниками програмного забезпечення, які інтегрують графові бази даних у свої додатки або здійснюють модернізацію існуючих систем. Їхня основна потреба полягає у простій та автоматизованій міграції даних, що дозволяє скоротити час розробки і забезпечити безперебійну інтеграцію з іншими сервісами.

Третя категорія включає адміністраторів баз даних, які відповідають за підтримку, оптимізацію та міграцію даних між різними СУБД. Їм важливо мати гнучкі налаштування міграції, можливість перевірки коректності трансформації та збереження продуктивності системи.

Остання категорія складається з аналітиків і фахівців з кібербезпеки, які використовують графові бази для пошуку зв’язків між сутностями, прогнозування ризиків та аналізу складних систем взаємодії. Їм необхідні точні алгоритми міграції, здатні обробляти складні зв’язки і зберігати їхню структуру без втрати критичної інформації.

Таким чином, розроблюваний продукт задовольняє потреби різних користувачів, забезпечуючи широкий функціонал і високий рівень автоматизації для ефективного переходу до графових моделей даних.

**2.4 General Constraints**

Серверна частина системи має бути розроблена за допомогою технології ASP.NET для хостингу на платформі Microsoft Azure.

Клієнтська частина система має бути розроблена за допомогою бібліотеки Angular 12.

Платформа повинна працювати на будь-якому пристрої, в будь-якому браузері, який підтримує стандарт HTML5. Усі обчислення та операції виконуються безпосередньо на віддаленому сервері.

Для розробки та розгортання API будуть використовуватися середа розробки Microsoft Visual Studio.

**2.5 Assumptions and Dependencies**

**Пр-1**: використання Microsoft SQL Server як джерела реляційних даних для міграції у графову базу даних Neo4j.

**Пр-2**: необхідність підтримки Neo4j, включаючи сумісність з версією Cypher та можливість оптимізації роботи з графовими запитами.

**Пр-3**: використання браузерів, які підтримують HTML5.

* **Зал-1**: можливість використовувати новітні функції та переваги мови розмітки, що дозволить покращити досвід користувачів.

**3. Specific Requirements**

**3.1 External Interface Requirements**

**3.1.1 User Interfaces**

Користувачі мають можливість здійснити міграцію даних використовуючи інтерфейс командного рядка, а також виконати запити на результуючих структурах у браузерах, що підтримують HTML5. Інтерфейс має відповідати наступним евристикам юзабіліті:

- гнучкість і ефективність;

- свобода дій;

- запобігання помилок.

**3.1.2 Hardware Interfaces**

Серверна частина може бути розгорнута на будь-якій машині, що має не менше 2 ГБ ОЗУ та не менше 1 ГБ вільного простору на жорсткому диску.

**3.1.3 Software Interfaces**

Серверна частина може бути розгорнута на будь-якій машині, на якій встановлено .NET8. У якості бази даних використовується MSSQL та Neo4J.

Для використання необхідно мати встановлений та правильно налаштований браузер з підтримкою HTML5.

**3.1.4 Communications Interfaces**

Комунікація між сервером та клієнтами здійснюється за допомогою протоколу HTTPS. Комунікація між сервером системи та сервером бази даних здійснюється за допомогою протоколу TCP/IP.

**3.2 Functional Requirements**

**3.2.1 Імпорт реляційних даних**

3.2.1.1 Introduction

Система забезпечує можливість імпорту даних з реляційної бази Microsoft SQL Server для їх подальшої міграції у графову модель даних Neo4j. Користувач може обрати таблиці та атрибути, які потрібно конвертувати, а також налаштувати параметри імпорту.

3.2.1.2 Inputs

Текст: рядок підключення до бази даних MSSQL.

3.2.1.3 Processing

Під час виконання імпорту система надсилає запити до MSSQL для отримання структурованих даних. Після отримання даних вони аналізуються на відповідність заданим параметрам та готуються до трансформації у графову модель.

3.2.1.4 Outputs

Якщо імпорт успішний, дані будуть підготовлені до міграції у графову базу даних та представлені у структурованому форматі. У випадку помилки користувач отримає повідомлення з її описом.

3.2.1.5 Error Handling

Повідомлення з помилкою з'явиться, якщо:

а) неможливо встановити зв’язок з MSSQL;

б) структура таблиць не відповідає очікуваному формату.

**3.2.2 Міграція даних у графову модель**

3.2.2.1 Introduction

Процес перетворення реляційних даних у графову структуру передбачає автоматичне визначення логічних зв’язків між сутностями та конвертацію таблиць у вузли і зв’язки графа.

3.2.2.2 Inputs

Дані, отримані з MSSQL після імпорту; правила перетворення; схема відповідності реляційної та графової моделі.

3.2.2.3 Processing

Система аналізує структуру реляційних таблиць та виконує відповідну трансформацію, використовуючи алгоритми міграції.

3.2.2.4 Outputs

Конвертовані дані у форматі графової бази Neo4j.

3.2.2.5 Error Handling

Повідомлення з помилкою з'явиться, якщо:

а) неможливо встановити зв’язок з Neo4j;

б) неможливо визначити зв’язки між сутностями.

**3.3 Use Cases**

**3.3.1 Взаємодія користувачів з системою**

Система призначена для автоматизованої міграції даних з реляційної бази Microsoft SQL Server у графову базу даних Neo4j. Взаємодія користувача з системою розпочинається з доступу до її інтерфейсу, де він може налаштувати параметри імпорту та перетворення даних.

Перед початком роботи користувач обирає таблиці та атрибути, які необхідно перенести, а також визначає логіку формування графових зв’язків. Після цього запускається процес імпорту, що виконує запити до MSSQL для отримання структурованих даних.

Користувач також має змогу перевірити коректність отриманої графової структури за допомогою запитів до вхідної та результуючої СУБД. Система автоматично порівняє результати отриманих запитів.

**3.4 Classes / Objects**

**3.4.1 Таблиця**

3.4.1.1 Attributes

* Name
* Columns
* PrimaryKey
* ForeignKeys

3.4.1.2 Functions

* Визначення атрибутів таблиці
* Встановлення первинного ключа
* Формування зовнішніх зв’язків між таблицями

**3.4.2 Запит**

3.4.2.1 Attributes

* QueryText
* ExecutionTime
* ResultSet

**3.5 Non-Functional Requirements**

**3.5.1 Performance**

Система повинна забезпечувати середній час виконання міграції даних, що не перевищує 10 секунд для набору даних середнього розміру (до 15 тис. записів). У разі перевищення цього часу користувач має отримати повідомлення про поточний стан міграції.

**3.5.2 Reliability**

Всі помилки, що виникають у процесі міграції, повинні бути автоматично зафіксовані у логах системи. Користувачеві необхідно надавати зрозумілі повідомлення про можливі проблеми, а система має гарантувати безперебійність роботи не менш ніж 98% часу у стандартних умовах експлуатації.

**3.5.3 Availability**

Система повинна бути доступна для користування цілодобово. У разі збоїв підключення або непередбачуваних ситуацій має бути реалізований механізм повторного виконання операцій без втрати попередньо завантажених даних.

**3.5.4 Security**

Система повинна забезпечувати перевірку всіх вхідних даних перед їхньою обробкою. Специфікація міграції даних має включати механізми захисту від SQL-ін’єкцій та несанкціонованого доступу до реляційної та графової бази даних. Дані користувача не повинні бути змінені чи втрачені у процесі перетворення.

**3.5.5 Maintainability**

Архітектура системи має бути модульною, що дозволить легко оновлювати алгоритми міграції, додавати нові правила трансформації даних та забезпечувати підтримку додаткових форматів структурованих даних.

**3.5.6 Portability**

Програмне забезпечення повинно підтримувати роботу як у Windows, так і Linux середовищах. Інтерфейс управління міграцією має бути доступний через веб-додаток або API, що дозволить інтегрувати систему у сторонні програмні рішення.

**3.6 Inverse Requirements**

Система не повинна змінювати або видаляти вихідні дані реляційної бази під час процесу міграції. Всі трансформації виконуються у графовій моделі без впливу на вихідні джерела..

**3.7 Design Constraints**

Алгоритм міграції повинен підтримувати адаптивне перетворення даних, враховуючи характеристики реляційної моделі. Інтерфейс має бути інтуїтивно зрозумілим, із можливістю перегляду етапів міграції в реальному часі.

**3.8 Logical Database Requirements**

Вся інформація про процес міграції повинна зберігатися у журналах системи, а графова база даних має підтримувати резервне копіювання з періодичністю раз на 12 годин.

**4. Analysis Models**

**4.1 Activity Diagrams**

A diagram of a computer program

AI-generated content may be incorrect.

**5. Change Management Process**

У разі зміни обсягу проекту або вимог SRS може бути оновлено скрам-майстром або старшими розробниками. Всі зміни повинні бути схвалені скрам-майстром та власником продукту.