Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

УДК 004.657:004.051:004.413.5
DOI: https://doi.org/10.30837/2522-9818.2025.3.045

Д. Михневич, О. Мазурова, О. Перепичай

МЕТОД GRAPHMIGR8 МІГРАЦІЇ РЕЛЯЦІЙНИХ ДАНИХ У ГРАФОВУ МОДЕЛЬ NEO4J

У сучасному світі оброблення інформації все більшої актуальності набувають графові бази даних, що дають змогу ефективно моделювати та обробляти складні взаємозв'язки між сутностями предметних галузей. Нині основою більшості наявних програмних систем залишаються реляційні бази даних. Проте ці бази даних не завжди є кращим рішенням для програмних систем, для яких гостро постають вимоги масштабованості та високої доступності інформації. З метою підвищення продуктивності таких систем на практиці все частіше приймаються рішення щодо міграції їх реляційних баз даних в *NoSQL*, зокрема в графові бази даних. **Предметом дослідження** статті є методи міграції структурованих даних з реляційної моделі баз даних у графову модель. **Мета роботи** – підвищити ефективність міграції реляційних баз даних у графові бази даних способом розроблення продуктивного, адаптованого для системи управління базами даних *Neo4j* методу міграції та надання рекомендацій щодо ефективного впровадження методів міграції в графові бази даних. У роботі розв'язуються такі **завдання:** створення логічних моделей для реляційної та графової БД *Neo4j* в різних предметних галузях для проведення на їх основі експериментів з міграції; розроблення методу міграції реляційних даних у графову модель *Neo4j*; планування та проведення експериментального дослідження ефективності запропонованого методу порівняно з іншими методами міграції, а також розроблення рекомендацій щодо особливостей їх використання. Упроваджено такі **методи:** проєктування баз даних; міграція в графові бази даних; експериментальне оцінювання продуктивності баз даних; розроблення, основане на системах управління базами даних *MS SQL Server* 18 та *Neo4j* 5.26 і середовищі розроблення *Visual Studio* 2022. **Досягнуті результати:** запропоновано метод міграції *GraphMigr8* реляційних даних у графову модель *Neo4j*; експериментально оцінено якість методів міграції за метриками продуктивності та семантичної цілісності інформації; сформовано рекомендації щодо використання методів міграції. **Висновки:** виявлено переваги й недоліки методів міграції реляційних даних у графову модель даних, запропоновано метод міграції *GraphMigr8* реляційної БД у графову модель *Neo4j*, який продемонстрував кращу продуктивність та вищу семантичну відповідність трансформованих даних.
Ключові слова: база даних; графова модель; метод міграції; реляційна модель; СУБД; *Neo4j*.

Вступ

Сучасний стан розвитку інформаційних технологій визначається постійним зростанням обсягів та складності структурованих даних. Традиційні реляційні бази даних (БД), незважаючи на їх тривалу історію використання, демонструють певні обмеження в роботі зі складнопов'язаними даними, які є основою для роботи в проблемних сферах, що з'являються та стрімко розвиваються в наш час, а саме:

– соціальні мережі та аналіз соціальних зв'язків [1];
– системи рекомендацій;
– біоінформатика та кібербезпека;
– ігрова аналітика тощо.

Також реляційні БД мають певні проблеми із забезпеченням масштабованості [2] програмних систем. Як зазначається в роботі [3], на фоні стрімкого збільшення обсягів взаємопов'язаної інформації традиційні реляційні БД демонструють обмеження, особливо в процесі оброблення складних рекурсивних запитів. Це викликає необхідність пошуку альтернативних підходів до зберігання та оброблення інформації та, відповідно, міграції реляційних даних до нових, більш ефективних *NoSQL* моделей БД [4].

Графові БД, як представники напряму *NoSQL*, останнім часом набувають усе більшої популярності завдяки здатності ефективно подавати складні взаємозв'язки між сутностями. Графи забезпечують природний спосіб подання взаємопов'язаної інформації, що суттєво спрощує навігацію по складних структурах [5]. Відповідно до звіту *db-engines* [6] графові системи управління базами даних (СУБД) входять до топ-10 та демонструють стійку тенденцію до зростання популярності.

Отже, графові БД є основою не тільки для створення нових програмних застосунків, але й моделлю, на яку спрямовані процеси міграції реляційних БД. Сучасні методи міграції не завжди дають однозначні результати, що потребує також верифікації БД після міграції.

Отже, актуальним і практично значущим є дослідження ефективності та вдосконалення наявних методів міграції реляційних даних у графову модель,

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2025. No. 3 (33)

що дасть змогу мінімізувати втрати інформації, прискорити процес міграції даних та забезпечити семантичну цілісність перетворення.

Аналіз проблеми й наявних методів

Порівняльний аналіз продуктивності реляційних і графових БД, проведений *Vicknair* у роботі [7], демонструє значні переваги графових БД в процесі оброблення складнопов'язаних даних. Графова БД – це механізм зберігання інформації, що поєднує базові графові структури вершин і ребер із технологією збереження та мовою запитів, що оптимізовані для зберігання та швидкого отримання щільно пов'язаних даних. На відміну від інших моделей, графові БД побудовані на концепції, що відношення між сутностями є такими самими важливими, як і самі сутності [8].

Незважаючи на загальну концепцію графових БД, реалізації графової моделі в напрямі *NoSQL*-систем можуть різнитися. Відповідно до цього будуть відрізнятися і методи міграції в ці графові моделі. Отже, для *NoSQL*-систем [4], які лише поєднують в собі загальні напрями графових, документо-орієнтованих тощо БД, але не надають уніфікованих моделей БД. Методи міграції необхідно обирати чітко під певну *NoSQL* СУБД.

Нині найбільш популярною в напрямі графових СУБД є *Neo4J* [9], яка має високу масштабованість і, як особливість, забезпечує повну підтримку атрибутів для вершин і ребер графа. *Neo4J* не є мультимодельною системою, а побудована для підтримки суто графової моделі, що забезпечує їй кращу продуктивність, а це є особливо важливим у міграції складної структурованої інформації.

Загальна методологія міграції даних передбачає такі етапи: аналіз вхідної структури даних, визначення правил відтворення вхідної структури в результуючу структуру, генерація результуючої структури та перенесення інформації до неї. Необхідно нагадати, що в роботі з *NoSQL*-системами етап перетворення вхідної структури у вихідну фактично відсутній. Наприклад, в *Neo4J*, як і в будь-якій графовій моделі, відсутнє поняття "схема даних", тому під час міграції інформації з реляційної БД етап створення схеми для графової БД відсутній, тобто не створюється певна абстрактна структура даних. Натомість рядки таблиць одразу перетворюються на певні вершини або ребра графа, тобто фактично відбувається етап перенесення даних.

Основною проблемою міграції інформації між різними моделями БД є перевірка якості міграції, а саме семантичної еквівалентності БД, адже структури вхідних та вихідних даних за такої міграції не збігаються. Найбільш ефективним варіантом перевірки якості міграції в такому разі є створення та виконання еквівалентних запитів до похідної БД та результуючої. Результати виконання запитів мають збігатися.

Серед методів міграції реляційних даних у графову модель доволі популярними є *Roberto De Virgilio* [10], *Yelda Ünal* [11] тощо. Окремо необхідно виокремити групу методів, що основані на *ER*-моделі БД та за своєю сутністю є більш схожими на методи логічного проєктування графових БД на основі *ER*-моделі [12]. Яскравим представником такого напряму є метод Чжихонг Нань та Сюе Бай [13]. Проте не варто вважати такі методи повноцінними представниками методів міграції між реляційною та графовою БД.

Доволі новим і перспективним методом міграції реляційної БД в графову є *Rel2Graph* [14]. Цей метод оснований на класифікації таблиць реляційної БД на таблиці сутностей і 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, 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 у зв'язку "багато до багатьох"). Далі відбувається перетворення таблиць у 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 в цьому разі стають атрибутами вершин. Таблиці зв'язування перетворюються на ребра графа: назва таблиці стає типом ребра, її рядки – безпосередньо ребрами, а атрибути перетворюються на атрибути ребер. Додатково в графі створюються ребра типу *HAS* для відтворення зв'язків "один до одного" та "один до багатьох".

Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Аналіз методу продемонстрував певні недоліки, які можуть призвести до неефективної міграції, а саме:

– таблиці, що мають два зовнішніх ключі та простий первинний ключ, визначаються як таблиці сутності, хоча дуже часто такі таблиці проєктуються саме для реалізації зв'язку "багато до багатьох", тобто мають лише три стовпці: первинний ключ (частіше за все генерується автоматично) та два зовнішніх ключі (посилання на дві таблиці, що зв'язуються);

– таблиці, що мають два зовнішніх ключі та складений первинний ключ, визначаються як таблиці зв'язування, хоча можливі таблиці сутностей, що мають два зовнішніх ключі та складений первинний ключ.

Також алгоритм приймає на вхід файл із *SQL*-скриптами для створення реляційної БД, а не під'єднується до наявної БД, витрачаючи в такий спосіб час на оброблення та парсинг текстового файлу.

Оскільки проаналізовані методи спрямовані на абстрактну міграцію даних з реляційної БД до графової без огляду на особливості тієї чи іншої графової моделі, то актуальним є адаптація наявних методів до певної СУБД, наприклад популярної *Neo4J*, удосконалення або створення нових методів для роботи саме з цією СУБД та дослідження їх ефективності.

Мета роботи й завдання

Мета статті – підвищити ефективність міграції реляційних баз даних у графові БД способом розроблення продуктивного, адаптованого для системи управління базами даних *Neo4j* методу міграції та надання рекомендацій щодо ефективного використання методів міграції в графові БД.

Дослідження потребує виконання таких завдань:

– створення логічних моделей для реляційної та графової БД *Neo4j* в різних предметних галузях для проведення на їх основі експериментів з міграції;

– розроблення методу міграції реляційних даних у графову модель *Neo4j*;

– планування та проведення експериментального дослідження ефективності запропонованого методу порівняно з іншими методами міграції та розроблення рекомендацій щодо особливостей їх упровадження.

Розв'язання завдань

A. Математичне моделювання

Для формалізації процесу міграції використано апарат теорії графів і взято до уваги реляційну БД та графову модель БД для *Neo4j*.

Розглянемо процес міграції реляційної БД D в графову БД G під керуванням *Neo4J*. Реляційну БД D можна подати у вигляді кортежу

$$D = \langle T, P, R \rangle, \quad (1)$$

де $T = \left\{ T_i \middle| i = \overline{1,n} \right\}$ – скінченна множина таблиць реляційної БД; $P = \left\{ P_i \middle| i = \overline{1,n} \right\}$ – скінченна множина атрибутів у БД, де $P_i = \left\{ P_{ij} \middle| j = \overline{1,m} \right\}$ – скінченна множина атрибутів таблиці T_i; $R \subseteq T \times T$ – множина зв'язків між таблицями.

Графова БД G під керуванням *Neo4j*, яка буде результатом міграції, може мати такий вигляд:

$$G = \langle V, E \rangle, \quad (2)$$

де V – множина вершин графа, $V = \left\{ V_l \left(PV_l \right) \middle| l = \overline{1,p} \right\}$, $V_l \left(PV_l \right)$ – скінченна множина вершин певного l-го типу, що визначається однаковою множиною атрибутів PV_l; $E \subseteq V \times V$ – множина ребер графа, $E = \left\{ E_k \left(PE_k \right) \middle| k = \overline{1,q} \right\}$, $E_k \left(PE_k \right)$ – скінченна множина ребер певного k-го типу, якій властива однакова множина атрибутів PE_k ребер.

Процес міграції можна подати як сукупність відтворень:

– $\varphi : T \to V \cup E$ – перетворення, що ставить у відповідність кожній таблиці $T_i \in T$ множину вершин $V_i \left(PV_i \right) \in V$ певного типу або множину ребер $E_i \left(PE_i \right) \in E$ певного типу;

– $\Psi : P \to PV \cup PE$ – перетворення, що ставить у відповідність кожній множині атрибутів $P_i \in P$ таблиці T_i множину PV_i атрибутів вершин або множину PE_i атрибутів ребер;

– $\Omega : R \to E$ – перетворення, що ставить у відповідність кожному зв'язку $\left(T_i, T_j \right) \in R$ множину ребер $\left(V_l, V_k \right) \in E$ певного типу.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online)
Innovative technologies and scientific solutions for industries. 2025. No. 3 (33)

Отже, для власного методу міграції реляційної БД у графову під керівництвом *Neo4J* необхідно визначити відтворення φ, Ψ та Ω, що задають правила перетворення таблиць, зв'язків і атрибутів реляційної БД в множини вершин і ребер певних типів графової БД та їх атрибути.

Б. Розроблення методу GraphMigr8 для міграції в Neo4J

У роботі, спираючись на метод *Rel2Graph* [14], запропоновано метод міграції *GraphMigr8* реляційної БД (1) у графову БД (2) під керівництвом *Neo4J*, що має на меті подолати недоліки попередніх методів і забезпечити більш гнучкий та ефективний процес міграції реляційних структур.

Особливістю запропонованого методу є більш однозначне й чітке виділення таблиць, що реалізують зв'язок типу "багато до багатьох". Зазвичай у сучасній практиці проєктування такі таблиці мають простий первинний ключ (сурогатний) та два зовнішніх ключі, що посилаються на відповідні таблиці, іноді з додатковим змістовним атрибутом. Альтернативний варіант структури такої таблиці зв'язування передбачає наявність двох зовнішніх ключів, що одночасно є складеним первинним ключем, також з можливим додатковим змістовним атрибутом. У методі *GraphMigr8* за наявності понад одного змістовного атрибута запропоновано розглядати такі таблиці як стрижневі, що моделюють сутності реального світу, а не зв'язки типу "багато до багатьох", та перетворювати їх під час міграції у вершини графової БД. Так, у методі *GraphMigr8* таблиця вважатиметься таблицею вершин певного типу, якщо для неї виконується одна з умов:

– зовнішні ключі відсутні;

– кількість зовнішніх ключів дорівнює одному або більше ніж двом;

– існує два зовнішніх ключі та простий первинний ключ, до того ж кількість атрибутів у таблиці перевищує чотири, тобто в наявності мінімум два змістовних атрибути;

– існує два зовнішніх ключі, що утворюють складений первинний ключ, до того ж кількість атрибутів у таблиці більше ніж три, тобто в наявності мінімум два змістовних атрибути.

Таблиця вважатиметься таблицею ребер певного типу, якщо для неї виконується одна з умов:

– існує два зовнішніх ключі та простий первинний ключ, до того ж кількість атрибутів не перевищує чотирьох, тобто є лише один змістовний атрибут;

– існує два зовнішніх ключі, що утворюють складений первинний ключ, у цьому разі кількість атрибутів не перевищує трьох, тобто може бути лише один змістовний атрибут.

На рис. 1 зображено блок-схему запропонованого методу міграції, що містить кінцеву послідовність кроків.

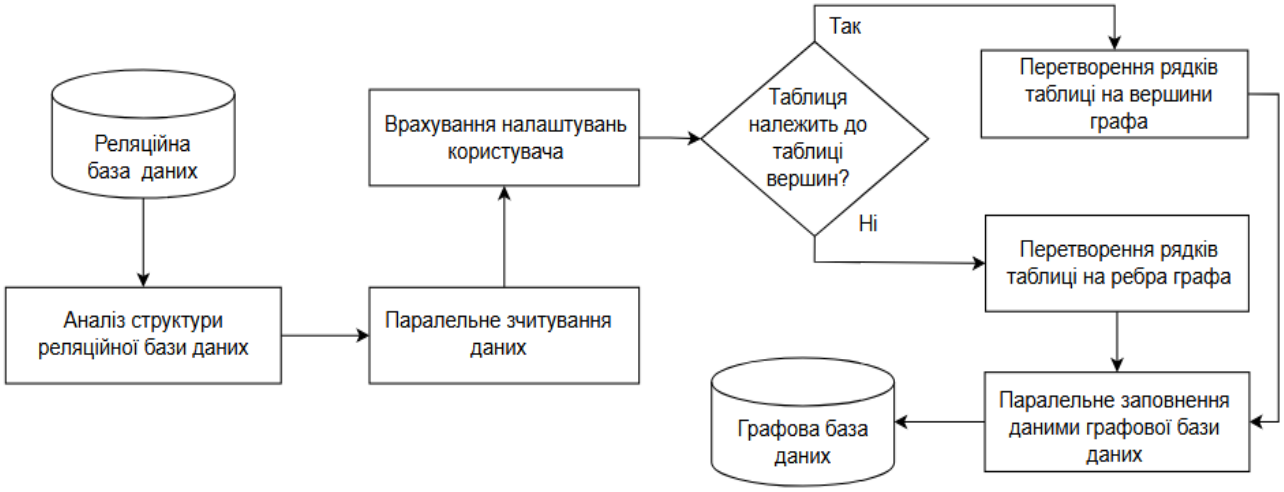

Рис. 1. Блок-схема методу міграції *GraphMigr8*

Розглянемо кроки запропонованого методу більш детально.

На кроці 1 для аналізу структури реальної реляційної БД здійснюється під'єднання безпосередньо до БД (в експериментальних дослідженнях під'єднання відбувалося до БД під керівництвом *Microsoft SQL Server*). Під час аналізу структури таблиць пропонується використовувати

Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

принципи, подані в роботі *Pokorný* [15], а саме під час аналізу система збирає детальну інформацію про кожну таблицю, зокрема:

– назви таблиці та атрибутів;
– тип даних атрибутів;
– первинні ключі;
– зовнішні ключі.

Зібрана інформація зберігається в таких структурах даних, як списки, геш-мережі та словники. У цьому разі для оптимізації алгоритму використовується паралельне зчитування інформації.

На кроці 2 беруться до уваги налаштування користувача, який може задати особливості предметної галузі.

Налаштування користувача дають змогу:

– визначати правила перетворення для специфічних таблиць;

– вилучати певні таблиці або атрибути з процесу міграції.

На кроці 3 відбувається класифікація таблиць за їх призначенням і структурою відповідно до наведених вище принципів.

На кроці 4 перетворюються рядки таблиць вершин у вершини графової БД. Назва кожної таблиці вершин стає типом вершини, а кожен її рядок перетворюється на окрему вершину.

Атрибути вершин формуються на основі атрибутів вхідної таблиці. Система зберігає всю семантичну інформацію, зокрема числові, текстові та інші типи інформації.

На кроці 5 перетворюються рядки таблиць ребер у ребра графа. Назва таблиці ребер стає типом ребра, а її рядки – безпосередньо ребрами між вершинами графа. Якщо в таблиці ребер були присутні змістовні (не ключові) атрибути, то вони перетворюються на атрибути ребер.

На етапі перетворення рядків таблиць ребер у ребра графа також створюються ребра типу *HAS* для всіх зовнішніх ключів, розташованих у таблицях вершин. Кожен зовнішній ключ стає ребром типу *HAS*, яке з'єднує поточну вершину (рядок таблиці із зовнішнім ключем) з відповідною цільовою вершиною (рядок таблиці, на який посилається зовнішній ключ).

На кроці 6 відбувається паралелізація перетворення та заповнення даними графової БД, яка є ключовим елементом підвищення продуктивності алгоритму. Для реалізації паралелізації використовуються такі підходи:

– паралельне зчитування даних – система розподіляє таблиці між декількома потоками, даючи змогу одночасно обробляти різні таблиці; кожен потік незалежно виконує аналіз структури та її класифікацію;

– паралельне заповнення графової БД даними – створення вершин та ребер відбувається паралельно; це досягається способом розподілу інформації на незалежні групи, які можуть оброблятися одночасно.

Отже, запропонований метод міграції *GraphMigr8* є гнучким інструментом для перетворення реляційних даних у графову БД.

B. Планування експериментального дослідження

Для експериментального дослідження продуктивності та ефективності запропонованого методу розглянуто декілька предметних галузей, для яких є доречним використання графових БД, зокрема сфери соціальних мереж, ігрових статистик та електронної комерції.

Розглянемо підготовку експериментального дослідження на прикладі розроблення БД для соціальної мережі. Ця сфера містить поняття про такі об'єкти реального світу, як користувачі, їхні профілі, зв'язки дружби, контент і взаємодії між користувачами. На рис. 2 зображено *ER*-діаграму БД у сфері соціальної мережі, що створено за нотацією *Richard Barker* із додатковими семантичними позначками зв'язків, що є запозиченням з нотації *Peter Pin-Shan Chen*.

Для оцінювання ефективності алгоритмів міграції обрано такі метрики:

– час трансформації даних (с) – час, який витрачає алгоритм на перетворення реляційної БД у графову; вимірювання часу для БД різних розмірів дав змогу оцінити масштабованість розроблених алгоритмів;

– обсяг використаної оперативної пам'яті (МВ) – кількісна оцінка обсягу пам'яті, що споживає алгоритм під час процесу міграції; ця метрика дала змогу порівняти ресурсоємність розроблених алгоритмів;

– відсоток збігу мігрованої структури із спроєктованою розробником (%); для чистоти дослідження під час упровадження методу *GraphMigr8* етап врахування налаштувань користувача було пропущено.

Важливим показником якості міграції даних є збереження їх семантичної цілісності [16]. Для оцінювання якості та повноти отриманих графових БД застосовано метрику семантичної

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2025. No. 3 (33)

еквівалентності запитів та їх результатів. Сутність цієї метрики полягає в порівнянні результатів, ідентичних за логікою запитів, сформованих для вхідної (реляційної) та вихідної (графової) БД.

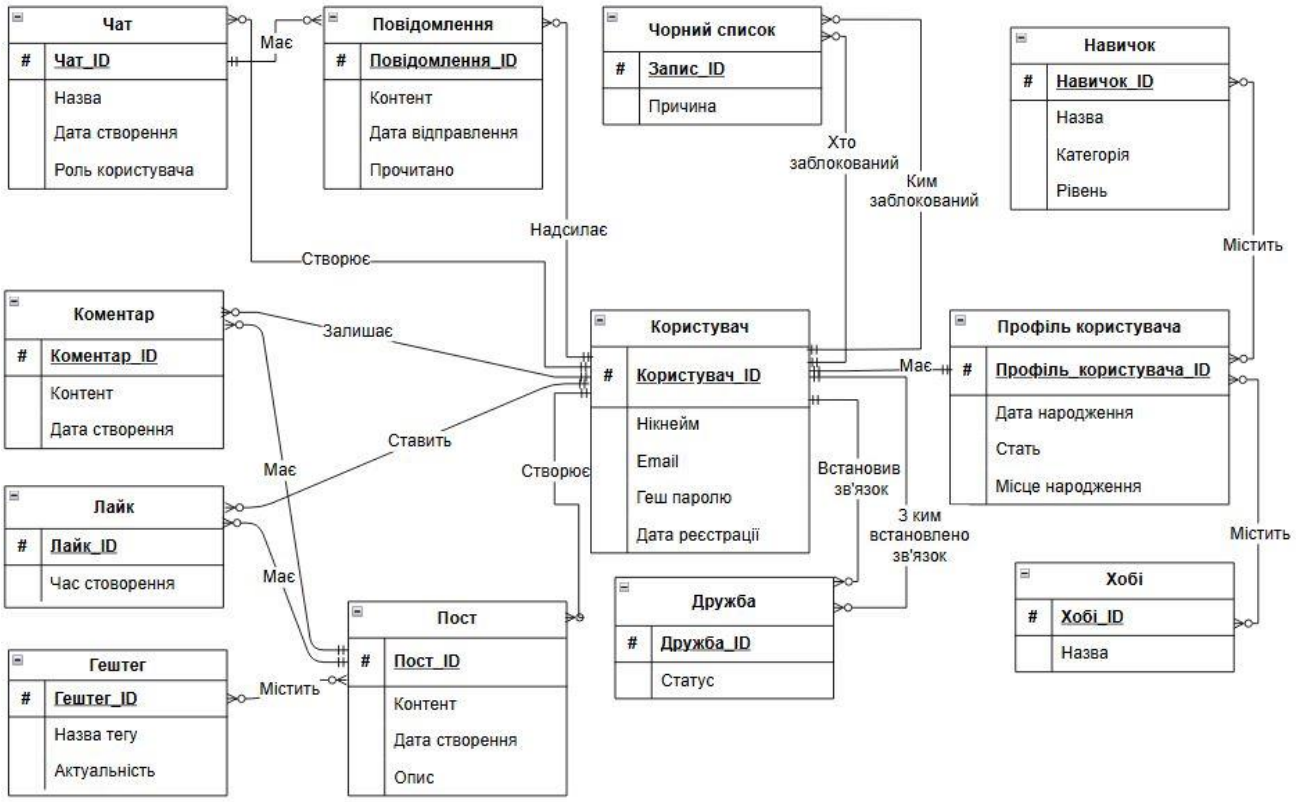

Рис. 2. *ER*-діаграма бази даних у сфері соціальної мережі

Використана для метрики технологія оцінювання передбачає такі дії:

– формування набору тестових запитів для вхідної реляційної БД;

– трансформацію цих запитів на мову *Cypher* для графової БД під керівництвом *Neo4j*;

– виконання запитів на базі СУБД *MS SQL Server ma Neo4J*, порівняння результатів тестових запитів за критерієм ідентичності кількості результуючих записів і значень за відповідними атрибутами.

Додатково для аналізу ефективності методів міграції оцінено продуктивність роботи з отриманими графовими БД способом виконання еквівалентних запитів на графових БД, створених різними алгоритмами міграції. Метриками оцінювання продуктивності роботи з графовими БД обрано:

– час виконання запитів (мс);

– завантаженість системних ресурсів (завантаженість процесора під час виконання запитів (%), обсяг споживання оперативної пам'яті (MB));

– ефективність зберігання інформації (загальний обсяг БД на диску (MB));

– кількість операцій доступу до БД, що виконуються під час виконання запиту (*DB hits*).

Результати досліджень та їх обговорення

З використанням описаних метрик порівняно два алгоритми методу *Rel2Graph* (у класичній імплементації та оптимізований алгоритм *Rel2Graph*, який використовує багатопотоковість) та розроблений алгоритм для *GraphMigr8*.

Експерименти проводилися на обчислювальній машині з такими характеристиками:

– процесор – Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz (загалом чотири ядра);

– оперативна пам'ять – 16 GB;

– 64-бітна операційна система.

Використовувалися СУБД таких версій: *MS SQL Server* 18 та *Neo4j* 5.26.

Більш детально розглянемо результати експериментів із базами даних середнього розміру, що було розроблено для більш адекватних щодо використання графових моделей сфер соціальних мереж та ігрових статистик, адже вони мають

Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

найбільш зв'язану структуру. У табл. 1 подано заміри за таким важливим показником ефективності, як час міграції реляційної БД у графову БД.

Оптимізований *Rel2Graph* та *GraphMigr8* демонструють значне скорочення часу міграції для обох БД завдяки оптимізації способом багатопотоковості. Метод *GraphMigr8* має найкращі показники у зв'язку із меншою кількістю результуючих вершин і ребер, що він утворює під час міграції.

Не менш важливим є вимірювання використаних системних ресурсів під час роботи алгоритмів (табл. 2).

Таблиця 1. *Час міграції (с)*

Алгоритм	Час міграції для БД *SocialNetwork* (с)	Час міграції для БД *GameDB* (с)
Rel2Graph	213.27	202.51
Rel2Graph оптимізований	8.71	15.24
GraphMigr8	7.93	5.5

Таблиця 2. *Використання системних ресурсів*

Алгоритм	Використання оперативної пам'яті для *SocialNetwork* (МБ)	Використання оперативної пам'яті для *GameDB* (МБ)	Використання процесора для *SocialNetwork* (%)	Використання процесора для *GameDB* (%)
Rel2Graph	6.7	9.83	14.31	13.99
Rel2Graph оптимізований	15.19	11.08	3.79	2.39
GraphMigr8	11.29	6.47	2.99	3.16

Хоча алгоритм *GraphMigr8* не показує очевидної переваги з погляду застосування оперативної пам'яті, але він значно знижує навантаження на процесор, що робить його більш ефективним щодо використання системних ресурсів, а особливо для систем, де *CPU* є обмеженим ресурсом.

Проаналізуємо досягнуті результати для різних обсягів реляційних БД. Великою вважатимемо базу даних *Ecommerce*, середніми – *SocialNetwork* та *GameDB* та малими – *TravelingDb, ReportDb, SportDb*. Для середніх та малих БД наведемо усереднені значення. Отримані метрики подано в табл. 3.

Таблиця 3. *Результати міграції для БД різного обсягу*

Тип БД	Алгоритм	Час міграції (с)	Використання оперативної пам'яті (МБ)	Використання процесора (%)
Мала	*Rel2Graph*	8.6	9.6	6.5
	Rel2Graph оптимізований	4.67	2.08	3.8
	GraphMigr8	3.51	1.3	1.88
Середня	*Rel2Graph*	207.89	8.23	14.15
	Rel2Graph оптимізований	11.98	13.14	3.09
	GraphMigr8	6.715	8.88	3.08
Велика	*Rel2Graph*	10829.44	91.1	26.04
	Rel2Graph оптимізований	433.21	52.99	7.44
	GraphMigr8	427.84	42.2	7.07

Оригінальний *Rel2Graph* демонструє яскраво виражену експоненційну залежність від розміру БД. Перехід від середньої до великої БД спричиняє зростання часу міграції в понад 52 рази (з 207.89 с до 10829.44 с). Оптимізований *Rel2Graph* значно зменшує експоненційне зростання, хоча залежність усе ще не є лінійною. Алгоритм *GraphMigr8* показує найбільш наближену до лінійної залежність із незначним кутом нахилу, що є оптимальним для масштабування. На великій БД оптимізований *Rel2Graph* та *GraphMigr8* показали майже однаковий час міграції. Це пов'язано з тим, що сфера *Ecommerce* не є природною для зберігання у графовій БД. Отже, реляційна схема цієї БД містить таблиці, які обома методами визначаються як таблиці вершин, що не дає змогу задіяти переваги методу *GraphMigr8*. Незважаючи на це, розроблений *GraphMigr8* показав не гірші результати, ніж метод *Rel2Graph* [14].

Для малих БД алгоритм *GraphMigr8* використовує на 86% менше пам'яті порівняно з оригінальним *Rel2Graph*. За умови збільшення розміру БД спостерігається зростання застосованої пам'яті в усіх алгоритмах, проте *GraphMigr8* завжди демонструє кращі показники.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) Innovative technologies and scientific solutions for industries. 2025. No. 3 (33)

Оригінальний *Rel2Graph* найбільш інтенсивно використовує процесор, особливо в роботі з великими БД. Оптимізований *Rel2Graph* та *GraphMigr8* демонструють приблизно однакове навантаження на процесор для середніх і великих БД. Алгоритм *GraphMigr8* має найменше навантаження на процесор під час роботи з малими БД.

Перейдемо до розгляду експериментів щодо визначення семантичної цілісності результату міграції.

Було розроблено низку запитів мовами *SQL* та *Cypher* (офіційна мова запитів *Neo4j*) [17] та виконано їх на реляційній БД під керівництвом *MS SQL Server* та графовій БД під керівництвом *Neo4j*. Результати запитів порівнювалися автоматизовано за допомогою розробленого програмного забезпечення.

Для перевірки семантичної еквівалентності даних використовувалися серії статистичних запитів на підрахунок кількості сутностей різного типу, запитів на фільтрацію сутностей за різними атрибутами, а також запитів на перевірку зв'язності сутностей.

Наприклад, для сфери соціальної мережі застосовувалися запити, що, зокрема, містили:

– запит № 1 – пошук постів користувачів, імена яких починаються з літери "A";

– запит № 2 – пошук гештегів із релевантністю понад 0.7;

– запит № 3 – отримання середнього рівня навичок користувачів;

– запит № 4 – пошук потенційних друзів (запит до декількох зв'язаних сутностей).

Для сфери ігрової серверної системи застосовувалися запити:

– запит № 1 – пошук деталей спорядження персонажа (запит до декількох зв'язаних сутностей);

– запит № 2 – отримання інформації про предмети певного типу, що наявні в інвентарі персонажів;

– запит № 3 – пошук нагород за квести з певним атрибутом;

– запит № 4 – історія отриманих персонажем предметів.

Усі розроблені запити для обох досліджених методів повернули однакову кількість записів з ідентичними значеннями за атрибутами, що дає змогу зробити висновок про семантичну цілісність мігрованих даних.

Метрики продуктивності отриманих графових БД за результатами виконання цих запитів зведені в табл. 4. В цій серії експериментів досліджувалась лише одна реалізація методу *Rel2Graph*, адже алгоритми *Rel2Graph* та оптимізований *Rel2Graph* створюють однакову структуру графової БД, тож їх порівняння з погляду продуктивності графової БД не має сенсу. Отже, порівнювалися саме методи *Rel2Graph* та *GraphMigr8*, а також оцінювалося покращення метрик продуктивності для методу *GraphMigr8*.

Таблиця 4. *Продуктивність запитів*

База даних	**Запит**	**Метод**	**Оперативна пам'ять (байти)**	**Час (мс)**	***CPU* (%)**	***DB hits***
SocialNetwork	запит № 1	*Rel2Graph*	4472	3	7	445
		GraphMigr8	4472	2	6.3	445
		Покращення	0 %	33.3 %	10.0 %	0 %
	запит № 2	*Rel2Graph*	45000	7	10.2	5331
		GraphMigr8	40608	5	8	3257
		Покращення	9.8 %	28.6 %	21.6 %	38.9 %
	запит № 3	*Rel2Graph*	13672	5	7.3	3669
		GraphMigr8	12944	3	7.9	2512
		Покращення	5.3 %	40.0 %	-8.2 %	31.4 %
	запит № 4	*Rel2Graph*	14352	3	11.7	998
		GraphMigr8	9592	2	7.9	426
		Покращення	33.2 %	33.3 %	32.5 %	57.3 %
GameDB	запит № 1	*Rel2Graph*	2536	2	7.2	834
		GraphMigr8	2104	2	4.6	758
		Покращення	17.0 %	0 %	36.1 %	9.1 %
	запит № 2	*Rel2Graph*	8592	3	7.1	617
		GraphMigr8	8584	2	7.4	533
		Покращення	0.1 %	33.3 %	-4.2 %	13.6 %
	запит № 3	*Rel2Graph*	360	1	8.2	227
		GraphMigr8	344	1	6.2	181
		Покращення	4.4 %	0 %	24.4 %	20.3 %
	запит № 4	*Rel2Graph*	2256	2	6.2	718
		GraphMigr8	2256	1	6.8	682
		Покращення	0 %	50.0 %	-9.7 %	5.0 %

ISSN 2522-9818 (print)
Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33) ISSN 2524-2296 (online)

Унаслідок упровадження методу *GraphMigr8* спостерігається значне покращення використання оперативної пам'яті й здебільшого високі результати за часом виконання. Загалом метод *GraphMigr8* демонструє нижчу завантаженість процесора.

Детальний аналіз графових БД під час експериментів засвідчив, що однією з ключових переваг методу *GraphMigr8* є отримання більш досконалої структури графа відповідно до кількості його вершин і ребер. Кількість вузлів і ребер результуючих графових БД наведено в табл. 5 та оцінено відсоток покращення в разі використання методу *GraphMigr8*.

Таблиця 5. *Кількість вершин та ребер*

База даних	Метод	Кількість вершин	Кількість ребер
SocialNetwork	*Rel2Graph*	6744	12388
	GraphMigr8	2850	8494
	Покращення	57.7%	31.4%
GameDB	*Rel2Graph*	7885	12879
	GraphMigr8	2717	7711
	Покращення	65.5%	40.1%

Зменшення кількості вершин та ребер у процесі застосування методу міграції *GraphMigr8* безпосередньо вплинуло на розмір БД. Розміри результуючих графових БД наведено в табл. 6 та показано відсоток покращення для методу *GraphMigr8*.

Таблиця 6. *Розмір бази даних*

База даних	Метод	Розмір (MB)
SocialNetwork	*Rel2Graph*	3.35
	GraphMigr8	2.42
	Покращення	27.8 %
GameDB	*Rel2Graph*	2.46
	GraphMigr8	1.67
	Покращення	32.1 %

Останньою серією експериментів була перевірка збігу автоматично мігрованої структури до спроєктованої розробником вручну, який мав змогу взяти до уваги особливості предметної галузі. Для цього вручну спроєктовано графові БД на основі *ER*-діаграм, на базі яких було розроблено відповідні реляційні БД для обраних сфер. Приклад *ER*-діаграми та спроєктованої вручну графової БД для сфери соціальної мережі зображено на рис. 2 та 3 відповідно.

Розглянемо деякі результати цієї серії експериментів.

Подамо назви таблиць реляційних БД для кращого розуміння результату міграції:

– *TravelingDb – Achievement, Role, User, UserAchievement;*

– *SocialNetwork – BlackList, Chat, Comment, Friendship, Hashtag, HashtagPost, Hobby, Like, Message, Post, Skill, User, UserChat, UserHobby, UserProfile, UserSkill;*

– *Ecommerce – Basket, Category, Order, Product, Product_basket, Product_order, Product_property, Property, User.*

Результати експерименту наведені в табл. 7.

Результати експериментів для БД малого розміру показали, що графові моделі, мігровані за допомогою методу *GraphMigr8*, мають більш ідентичну структуру до БД, що були спроєктовані вручну. Натомість графові моделі, мігровані за допомогою методу *Rel2Graph*, менш ідентичні до спроєктованих вручну (наприклад, БД *TravelingDb*).

БД середнього розміру для сфер соціальних мереж та ігрових серверних застосунків, які визначаються високим ступенем взаємозв'язків між сутностями, є найбільш прийнятними кандидатами для використання графових БД і, відповідно, є більш показовими об'єктами для експериментальних досліджень. Як бачимо, для БД *SocialNetwork* метод *GraphMigr8* продемонстрував більш компактну графову структуру, на відміну від спроєктованої вручну. Так, таблиця *Like* була перетворена на ребра між вершинами *User* та *Post*, що привело до більш компактної та логічної структури графа. Для БД *GameDB* спроєктована вручну графова структура виявилася більш компактною. Але метод *GraphMigr8* знову продемонстрував кращі результати, ніж *Rel2Graph*.

Для БД *Ecommerce* графові структури для всіх трьох методів виявилися ідентичними, якщо не брати до уваги назви ребер. Це можна пояснити тим, що всі проміжні таблиці (таблиці зв'язування) в цій реляційній БД мають складений первинний ключ, частини якого одночасно є зовнішніми ключами, та кількість змістовних атрибутів не перевищує двох. У такому разі методи *Rel2Graph* та *GraphMigr8* створюють однакові графові структури. До того ж необхідно зазначити, що розглянутий фрагмент сфери електронної комерції майже ніколи не проєктується розробниками у вигляді графової БД. Отже, це порівняння не можна вважати показовим.

ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Таблиця 7. *Результати порівняння автоматично мігрованої структури із спроєктованою розробником*

База даних	Метод	Кількість вершин	Кількість ребер	Назви вершин	Назви ребер
TravelingDb	Ручне проєктування	44	112	Achievement, Role, User	Has_Achievements, Has_Role
	Rel2Graph	136	204	Achievement, Role, User, UserAchievement	HAS_Achievement _UserAchievement, HAS_Role_User, HAS_User_UserAchievement
	GraphMigr8	44	112	Achievement, Role, User	HAS_Role_User, UserAchievement
SocialNetwork	Ручне проєктування	4813	10457	Chat, Comment, Hashtag, Hobby, Like, Message, Post, Skill, User, UserProfile	HAS (чотири зв'язки), Send, Leaves, Puts, Creates, Friends_with, Has_blocked, Contains (три зв'язки)
	Rel2Graph	6744	12388	BlackList, Chat, Comment, Friendship, Hashtag, HashtagPost, Hobby, Like, Message, Post, Skill, User, UserChat, UserHobby, UserProfile, UserSkill	HAS_Chat_Message, HAS_Chat_UserChat, HAS_Hashtag_HashtagPost, HAS_Hobby_UserHobby, HAS_Post_Comment, HAS_Post_HashtagPost, HAS_Post_Like, HAS_Skill_UserSkill, HAS_UserProfile_UserHobby, HAS_UserProfile_UserSkill, HAS_User_BlackList, HAS_User_Comment, HAS_User_Friendship, HAS_User_Like, HAS_User_Message, HAS_User_Post, HAS_User_UserChat, HAS_User_UserProfile
	GraphMigr8	2850	8494	Chat, Comment, Hashtag, Hobby, Message, Post, Skill, User, UserProfile	BlackList, Friendship, HAS_Chat_Message, HAS_Post_Comment, HAS_User_Comment, HAS_User_Message, HAS_User_Post, HAS_User_UserProfile, HashtagPost, Like, UserChat, UserHobby, UserSkill
Ecommerce	Ручне проєктування	1800098	9400041	Basket, Category, Order, Product, Property, User	HAS_Order, HAS_Basket, Placed_in, HAS_Property, Ordered, HAS_Parent
	Rel2Graph	1800098	9400041	Basket, Category, Order, Product, Property, User	HAS_Category_Category, HAS_Category_Product, HAS_User_Basket, HAS_User_Order, Product_basket, Product_order, Product_property
	GraphMigr8	1800098	9400041	Basket, Category, Order, Product, Property, User	HAS_Category_Category, HAS_Category_Product, HAS_User_Basket, HAS_User_Order, Product_basket, Product_order, Product_property

Проведена серія експериментів демонструє, що БД із значною кількістю простих зв'язків (без додаткових змістовних атрибутів) отримують більший ефект від міграції до графової моделі за допомогою алгоритму *GraphMigr8*. Натомість у системах електронної комерції, де зв'язки містять багато власних змістовних атрибутів, методи *GraphMigr8* та *Rel2Graph* демонструють майже ідентичні результати. У разі коли проміжні таблиці мають простий первинний ключ та два зовнішніх ключі, алгоритм *GraphMigr8* повертатиме більш компактну графову структуру.Отже, проведені експерименти продемонстрували суттєві переваги методу *GraphMigr8* до міграції реляційних БД у графові під керівництвом *Neo4J* , якщо порівнювати з методом *Rel2Graph*, особливо для предметних галузей із високою щільністю взаємозв'язків між сутностями, де графова модель даних є природно більш відповідною для подання та оброблення інформації. Наголосимо, що дослідження підтверджують ефективність запропонованого методу *GraphMigr8* міграції реляційних БД у графові БД *Neo4J*, демонструючи його переваги за ключовими показниками продуктивності.

Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Висновки й перспективи подальшого дослідження

У роботі запропоновано метод міграції реляційних БД у графову БД *Neo4j* - *GraphMigr8* та експериментально перевірено його ефективність.

Одним із ключових етапів роботи стало проєктування логічних моделей предметних галузей, що дало змогу створити ефективну основу для дослідження міграції. Зокрема розроблено моделі для БД соціальних мереж та ігрових серверів, які є прикладами доцільності міграції з реляційних БД у графові.

Досягнуті результати експериментів демонструють суттєві переваги методу *GraphMigr8*, а саме: скорочення часу міграції, більш ефективне використання системних ресурсів і зниження складності отриманих графових структур. Було сформульовано рекомендації щодо впровадження розглянутих методів міграції залежно від характеру та складності вхідних показників.

Розроблений метод *GraphMigr8* має певні обмеження, оскільки він розроблявся спеціально під міграцію даних до *Neo4j*, де у ребер є можливість зберігати атрибути. Проте не всі графові моделі підтримують цю функціональність. А втім метод *GraphMigr8* без суттєвих змін підійде для міграції в такі графові СУБД, як *Amazon Neptune, OrientDb, TigerGraph*, що також підтримають атрибути у ребер.

Натомість для СУБД *Apache Giraph* або *FlockDB*, де концепція атрибутів ребер обмежена або відсутня, застосування методу *GraphMigr8* потребуватиме модифікації. У цьому разі більш ефективним буде впровадження методу *Rel2Graph*.

Надалі доцільними є додаткові експериментальні дослідження з вищезгаданими СУБД для більш точного аналізу ефективності запропонованого методу в різних технологічних середовищах. Але можна зробити прогноз, що якісні переваги методу *GraphMigr8* збережуться порівняно з методом *Rel2Graph*, хоча ефективність кількісних показників може дещо змінитися. Ці розбіжності можуть бути зумовлені особливостями реалізації відповідних СУБД, специфікою їх мов запитів, оптимізацією внутрішніх алгоритмів оброблення інформації та ефективністю роботи з різними типами структур.

Список літератури

1. Vyawahare H. Exploring the Hybrid Approach: Integrating Relational and Graph Databases for Enhanced Data Management. *Communications in Computer and Information Science*. 2024. Vol. 2235, P. 176–191. DOI: https://doi.org/10.1007/978-3-031-74701-4_13
2. Yedilkhan D., Mukasheva A., Bissengaliyeva D. Performance Analysis of Scaling NoSQL vs SQL: A Comparative Study of MongoDB, Cassandra, and PostgreSQL. *IEEE International Conference on Smart Information Systems and Technologies* (SIST). 2023. P. 479–483. DOI: https://doi.org/10.1109/SIST58284.2023.10223568
3. Ragunathan A., Bhavani K., Sasi A. Integration of NoSQL and Relational Databases for Efficient Data Management in Hybrid Cloud Architectures. *Journal of Machine and Computing*. 2025. P. 1277–1287. DOI: https://doi.org/10.53759/7669/jmc202505100
4. Kuzochkina A., Shirokopetleva M., Dudar Z. Analyzing and Comparison of NoSQL DBMS. *International Scientific-Practical Conference Problems of Infocommunications. Science and Technology* (PIC S&T). 2018. P. 560–564. DOI: https://doi.org/10.1109/INFOCOMMST.2018.8632133
5. Mhedhbi A., Deshpande A., Salihoğlu S. Modern Techniques for Querying Graph-structured Databases. *Foundations and Trends® in Databases*. 2024. №14(2). P. 72–185. DOI: https://doi.org/10.1561/1900000090
6. DB-Engines Ranking. URL: https://db-engines.com/en/ranking (дата звернення: 25.12.2024)
7. Vicknair C., Macias M., Zhao Z. A comparison of a graph database and a relational database: a data provenance perspective. *Proceedings of the 48th Annual Southeast Regional Conference*, ACM. 2010. P.1–6. DOI: https://doi.org/10.1145/1900008.1900067
8. Bonifati A., Özsu M. T., Tian Y. The Future of Graph Analytics. *Companion of the 2024 International Conference on Management of Data* (SIGMOD '24). 2024. P. 544–545. DOI: https://doi.org/10.1145/3626246.3658369
9. Neo4j, Inc. How to create a graph data model. URL: https://neo4j.com/docs/model/ (дата звернення: 01.04.2025)
10. De Virgilio R., Maccioni A., Torlone R. Model-driven design of graph databases. *International Conference on Conceptual Modeling*, Springer. 2024. P.172-185. DOI: https://doi.org/10.1007/978-3-319-12206-9_14
11. Ünal Y., Oğuztüzün H. Migration of Data from Relational Database to Graph Database. *Proceedings of the 2018 International Conference on Information Systems and Computer Science*. 2018. P. 1–5. DOI: https://doi.org/10.1145/3200842.3200852

ISSN 2522-9818 (print)
ISSN 2524-2296 (online) *Innovative technologies and scientific solutions for industries. 2025. No. 3 (33)*

12. Mazurova O., Syvolovskyi I., Syvolovska O. NOSQL database logic design methods for MONGODB and NEO4J. *Innovative technologies and scientific solutions for industries*. 2022. № 2 (20). С. 52–63. DOI: https://doi.org/10.30837/ITSSI.2022.20.052
13. Nan Z., Bai X. The study on data migration from relational database to graph database. *Journal of Physics: Conference Series*. 2019. №1345(2), P. 022061. DOI: https://doi.org/10.1088/1742-6596/1345/2/022061
14. Zhao Z., Liu W., French T. Rel2Graph: Automated Mapping From Relational Databases to a Unified Property Knowledge Graph. *Research Square* [Preprint]. 2024. DOI: https://doi.org/10.21203/rs.3.rs-5451592/v1
15. Pokorný J. Graph Databases: Their Power and Limitations. *IFIP International Conference on Computer Information Systems and Industrial Management*. 2015. P. 58–69. DOI: https://doi.org/10.1007/978-3-319-24369-6_5
16. Jouili S., Vansteenberghe V. An empirical comparison of graph databases. *Proceedings of the 2013 International Conference on Social Computing*, IEEE. 2013. P.708–715. DOI: https://doi.org/10.1109/SocialCom.2013.106
17. Neo4j, Inc. The Neo4j Cypher Manual. URL: https://neo4j.com/docs/cypher-manual/current/ (дата звернення: 08.04.2025).

References

1. Vyawahare, H. (2024), "Exploring the Hybrid Approach: Integrating Relational and Graph Databases for Enhanced Data Management", *Communications in Computer and Information Science*, No. 2235, P. 176–191. DOI: https://doi.org/10.1007/978-3-031-74701-4_13
2. Yedilkhan, D., Mukasheva, A., Bissengaliyeva, D. and Suynullayev, Y. (2023), "Performance Analysis of Scaling NoSQL vs SQL: A Comparative Study of MongoDB, Cassandra, and PostgreSQL", *IEEE International Conference on Smart Information Systems and Technologies* (SIST), P. 479–483. DOI: https://doi.org/10.1109/SIST58284.2023.10223568
3. Ragunathan, A., Bhavani, K., Sasi, A. and Kumar, S. R. (2025), "Integration of NoSQL and Relational Databases for Efficient Data Management in Hybrid Cloud Architectures", *Journal of Machine and Computing*, P. 1277–1287. DOI: https://doi.org/10.53759/7669/jmc202505100
4. Kuzochkina, A., Shirokopetleva, M., Dudar, Z., (2018), "Analyzing and Comparison of NoSQL DBMS", *International Scientific-Practical Conference Problems of Infocommunications. Science and Technology* (PIC S&T), P. 560–564. DOI: https://doi.org/10.1109/INFOCOMMST.2018.8632133
5. Mhedhbi, A., Deshpande, A. and Salihoğlu, S. (2024), "Modern Techniques for Querying Graph-structured Databases", *Foundations and Trends® in Databases*, No.14(2), P. 72–185. DOI: https://doi.org/10.1561/1900000090
6. DB-Engines Ranking, available at: https://db-engines.com/en/ranking (last accessed 25.12.2024)
7. Vicknair, C., Macias, M., Zhao, Z., Nan, X., Chen, Y. and Wilkins, D. (2010), "A comparison of a graph database and a relational database: a data provenance perspective", *Proceedings of the 48th Annual Southeast Regional Conference*, ACM, P.1–6. DOI: https://doi.org/10.1145/1900008.1900067
8. Bonifati, A., Özsu, M. T., Tian, Y., Voigt, H., Yu, W. and Zhang, W. (2024), "The Future of Graph Analytics", *Companion of the 2024 International Conference on Management of Data* (SIGMOD '24), P. 544–545. DOI: https://doi.org/10.1145/3626246.3658369
9. Neo4j, Inc. How to create a graph data model, available at: https://neo4j.com/docs/model/ (last accessed 01.04.2025).
10. De Virgilio, R., Maccioni, A. and Torlone, R. (2014), "Model-driven design of graph databases", *International Conference on Conceptual Modeling*, Springer, P.172–185. DOI: https://doi.org/10.1007/978-3-319-12206-9_14
11. Ünal, Y. and Oğuztüzün, H. (2018), "Migration of Data from Relational Database to Graph Database", *Proceedings of the 2018 International Conference on Information Systems and Computer Science*, P. 1–5. DOI: https://doi.org/10.1145/3200842.3200852
12. Mazurova, O., Syvolovskyi, I. and Syvolovska, O. (2022), "NOSQL database logic design methods for MONGODB and NEO4J", *Innovative technologies and scientific solutions for industries*, No. 2(20), P. 52–63. DOI: https://doi.org/10.30837/ITSSI.2022.20.052
13. Nan, Z. and Bai, X. (2019), "The study on data migration from relational database to graph database", *Journal of Physics: Conference Series*, No. 1345(2), P. 022061. DOI: https://doi.org/10.1088/1742-6596/1345/2/022061
14. Zhao, Z., Liu, W. and French, T. (2024), "Rel2Graph: Automated Mapping From Relational Databases to a Unified Property Knowledge Graph", *Research Square* [Preprint]. DOI: https://doi.org/10.21203/rs.3.rs-5451592/v1
15. Pokorný, J. (2015), "Graph Databases: Their Power and Limitations", *IFIP International Conference on Computer Information Systems and Industrial Management*, P. 58–69. DOI: https://doi.org/10.1007/978-3-319-24369-6_5
16. Jouili, S. and Vansteenberghe, V. (2013), "An empirical comparison of graph databases", *Proceedings of the 2013 International Conference on Social Computing*, IEEE, P.708–715. DOI: https://doi.org/10.1109/SocialCom.2013.106
17. Neo4j, Inc. The Neo4j Cypher Manual, available at: https://neo4j.com/docs/cypher-manual/current/ (last accessed 08.04.2025).

Надійшла (Received) 19.04.2025

Сучасний стан наукових досліджень та технологій в промисловості. 2025. № 3 (33)
ISSN 2522-9818 (print)
ISSN 2524-2296 (online)

Відомості про авторів / About the Authors

Михневич Дмитро Костянтинович – магістр, Харківський національний університет радіоелектроніки, спеціальність 121 – Інженерія програмного забезпечення, Харків, Україна; email: dmytro.mykhnevych@nure.ua; ORCID ID: https://orcid.org/0009-0001-4854-6526

Мазурова Оксана Олексіївна – кандидат технічних наук, доцент, Харківський національний університет радіоелектроніки, доцент кафедри програмної інженерії, Харків, Україна; email: oksana.mazurova@nure.ua; ORCID ID: https://orcid.org/0000-0003-3715-3476

Перепичай Олег Іванович – аспірант, Харківський національний університет радіоелектроніки, спеціальність 121 – Інженерія програмного забезпечення, Харків, Україна; email: oleh.perepychai@nure.ua; ORCID ID: https://orcid.org/0009-0007-6931-7769

Mykhnevych Dmytro – Master, Kharkiv National University of Radio Electronics, Master of Specialty 121 – Software Engineering, Kharkiv, Ukraine.

Mazurova Oksana – PhD (Engineering Sciences), Associate Professor, Kharkiv National University of Radio Electronics, Associate Professor at the Department of Software Engineering, Kharkiv, Ukraine.

Perepichai Oleg – Graduate Student, Kharkiv National University of Radio Electronics, Graduate Student of Specialty 121 – Software Engineering, Kharkiv, Ukraine.

METHOD GRAPHMIGR8 FOR MIGRATING RELATIONAL DATA TO THE NEO4J GRAPH MODEL

In the modern world of data processing, graph databases are becoming increasingly relevant, allowing for efficient modeling and processing of complex relationships between entities in subject domains. Today, relational databases remain the foundation of most existing software systems. However, relational databases are not always the best solution for software systems that face stringent requirements for scalability and high data availability. In the direction of improving the performance of such systems, decisions regarding migration of their relational databases to NoSQL, particularly to graph databases, are increasingly being made in practice. **The subject matter** of the article is methods for migrating structured data from the relational database model to the graph model. **The goal** of the work is to improve the efficiency of migrating relational databases to graph databases by developing a productive method of migration adapted for the Neo4j database management system and providing recommendations for the effective use of migration methods to graph databases. The work addresses the following **tasks**: development of logical models for relational and graph databases Neo4j in various subject areas for conducting experiments on migration based on them; development of a method for migrating relational data to the Neo4j graph model; planning and conducting experimental research on the effectiveness of the proposed method compared to other migration methods, and development of recommendations regarding the peculiarities of their use. The following **methods** are used: database design methods; migration methods to graph databases; methods for experimental evaluation of database performance; development methods based on MS SQL Server 18 and Neo4j 5.26 database management systems, Visual Studio 2022 development environment. The following **results** were obtained: the GraphMigr8 method for migrating relational data to the Neo4j graph model was proposed; experimental evaluation of the quality of migration methods according to performance metrics and semantic data integrity was conducted; recommendations for using migration methods were formed. **Conclusions:** strengths and weaknesses of existing methods for migrating relational data to the graph data model were identified, the GraphMigr8 method for migrating relational databases to the Neo4j graph model was proposed, which showed better performance and higher semantic correspondence of transformed data.

Keywords: database; graph model; migration method; relational model; DBMS; Neo4j.

Бібліографічні описи / Bibliographic descriptions

Михневич Д.К., Мазурова О.О., Перепичай О.І. Метод GRAPHMIGR8 міграції реляційних даних у графову модель NEO4J. *Сучасний стан наукових досліджень та технологій в промисловості.* 2025. № 3 (33). С. 45–57. DOI: https://doi.org/10.30837/2522-9818.2025.3.045

Mykhnevych, D., Mazurova, O., Perepichai, O. (2025), "Method GRAPHMIGR8 for migrating relational data to the NEO4J graph model", *Innovative Technologies and Scientific Solutions for Industries*, No. 3 (33), P. 45–57. DOI: https://doi.org/10.30837/2522-9818.2025.3.045