УДК 004.652

Н.О. СОЛОДКА, Є.О. ПОЛІЩУК, О.А. ЛЯШЕНКО
Державний вищий навчальний заклад
«Український державний хіміко-технологічний університет», м. Дніпро

ВИКОРИСТАННЯ ГРАФОВОЇ ТА РЕЛЯЦІЙНОЇ МОДЕЛЕЙ ДАНИХ ПРИ РОЗРОБЦІ ЕКСПЕРТНИХ СИСТЕМ

Виконано порівняння ефективності застосування реляційної (MySQL) та нереляційної (Neo4j) систем управління базами даних для побудови бази знань експертних систем з використанням семантичної мережі. У якості графової системи управління базами даних розглянуто найбільш відомого представника таких систем – Neo4j з власною мовою запитів Cypher. Графові бази даних добре застосовні для аналізу зв'язків, тому великий інтерес представляє використання графових баз даних для створення експертних систем, для яких природньою є графова структура даних.

На прикладах формування певних запитів з використанням двох різних моделей даних, з'ясовано, чи є графова модель більш доцільним рішенням при проектуванні та створенні семантичних мереж, де потрібно зберігати та обробляти складні ієрархічні зв'язки між об'єктами.

Для порівняння ефективності двох систем управління базами даних використані наступні показники: певний набір запитів (мовами MySQL та Cypher), швидкість їх обробки та простота розуміння запитів.

При розглянутому обсязі даних, достатньому для формування експертної системи зроблено висновок, що виявлення переваг графової бази даних, таких як високі показники швидкості роботи запитів та лаконічність коду не відбувається.

Аналіз виконання тестів продемонстрував, що обидві системи управління базами даних впорались майже однаково. Час пошуку в реляційній і графовій базі даних відрізняється не суттєво. Складнощі при формуванні запитів в реляційній базі даних можна уникнути зокрема за рахунок нестандартних алгоритмічних рішень.

Ключові слова: бази даних, графова модель, запит, ієрархічні зв'язки, модель даних, мова запитів Cypher, реляційна модель, семантична мережа, структури даних, Java, Neo4j.

Н.А. СОЛОДКАЯ, Е.А. ПОЛИЩУК, О.А. ЛЯШЕНКО
Государственное высшее учебное заведение
«Украинский государственный химико-технологический университет», г. Днипро

ИСПОЛЬЗОВАНИЯ ГРАФОВОЙ И РЕЛЯЦИОННОЙ МОДЕЛЕЙ ДАННЫХ ПРИ РАЗРАБОТКЕ ЭКСПЕРТНЫХ СИСТЕМ

Выполнено сравнение эффективности применения реляционной (MySQL) и нереляционной (Neo4j) систем управления базами данных для построения базы знаний экспертных систем с использованием семантической сети. В качестве графовой системы управления базами данных рассмотрен наиболее известный представитель таких систем – Neo4j с собственным языком запросов Cypher. Графовые базы данных хорошо применимы для анализа связей, поэтому большой интерес представляет использование графовых баз данных для создания экспертных систем, для которых естественной является графовая структура данных.

На примерах формирования определенных запросов с использованием различных моделей данных, выяснено, является графовая модель более целесообразным решением при проектировании и создании семантических сетей, где нужно хранить и обрабатывать сложные иерархические связи между объектами.

Для сравнения эффективности двух систем управления базами данных использованы следующие показатели: определенный набор запросов (на языках MySQL и Cypher), время выполнения для заданного размера записей (скорость их обработки), простота понимания и программной реализации запросов. При рассматриваемом объеме данных, достаточном для формирования экспертной системы, можно сделать вывод, что выявление преимуществ графовой базы данных, таких как высокие показатели скорости работы запросов и лаконичность кода не происходит.

Анализ тестов показал, что обе системы управления базами данных справились почти одинаково. Время поиска в реляционной и графов базе данных отличается не существенно Сложности при формировании запросов в реляционной базе данных можно избежать в том числе за счет нестандартных алгоритмических решений.

Ключевые слова: базы данных, графова модель, запрос, иерархические связи, модель данных, реляционная модель, семантическая сеть, структуры данных, язык запросов Cypher, Java, Neo4j.

N.O. SOLODKA, E.O. POLISHYK, O.A. LIASHENKO
Ukrainian State University of Chemical Technology, Dnipro

USING THE GRAPH DATABASE MODEL AND THE RELATIONAL MODEL WHILE DEVELOPING EXPERT SYSTEMS

Comparison of the effectiveness of the use of relational (MySQL) and non-relational (Neo4j) database management systems for building an expert systems knowledge base. The semantic network is chosen as a model of knowledge representation models. As the graph database management system, the most prominent representative of such systems – Neo4j with its own language of queries Cypher – is considered.Graph databases are well suited for analyzing interconnections, which is why there has been a lot of interest in using graph databases to create expert systems.

Using the examples of forming certain requests using two data models, it was found out that the graph model is a more expedient solution for designing and creating semantic networks where you need to store and process complex hierarchical connections between objects.

To compare the effectiveness of the two database management systems, the following indicators were used: the certain query set (in MySQL and Cypher languages), execution time for a given entries size, ease of understanding and software implementation requests.

With the considered volume of data sufficient to form an expert system, it can be concluded that the identification of the advantages of a graph database, such as high rates of query performance and code brevity does not occur.

Analysis of the tests showed that both database management systems performed handled almost the same. Execution time in the relational database and graph database is not significantly different. Difficulties in forming queries in a relational database can be avoided, including through non-standard algorithmic solutions.

Keywords: database, graph model, query, hierarchical relationships, data model, relational model, semantic network, data structures, query language Cypher, Java, Neo4j.

Постановка проблеми

Найбільш поширеною моделлю представлення знань в експертних системах (ЕС) є семантична мережа, як найбільш наочна та доступна для програмної реалізації. Семантичні мережі зображуються у вигляді орієнтованого графа або дерева, яке має кореневий вузол та вузли без нащадків.

Не зважаючи на те, що з 1980-х рр. реляційні моделі даних посідають домінуюче положення серед засобів збереження даних, з часом перед розробниками програмних додатків стали виникати завдання, для вирішення яких реляційні бази даних (БД) не завжди дають задовільні результати в плані швидкості виконання запитів та простоти відображення структури даних [1]. Більш того, іноді програмна реалізація запитів мовою SQL є складним питанням. Наприклад, якщо є необхідність реалізувати багаторівневі зв’язки в реляційній БД, то це буде багаторівневий ланцюг складних запитів на основі з’єднань join.

Традиційно в реляційній базі дані зберігаються в стовпцях і рядках. Тим не менш, стовпці та рядки насправді це не так, як дані існують в реальному світі. Скоріше, дані існують як об'єкти та відносини між ними [1]. Саме в ЕС відносини між даними часто можуть бути більш цінними, ніж самі дані. А реляційна модель виявляється не досить гнучкою для створення складних запитів, обробки різноманітних ієрархічних і багаторівневих зв’язків між об’єктами.

Як альтернатива реляційних баз даних SQL з початку 2000-х набули свого розвитку так звані NoSQL (Not only SQL, не тільки SQL) сховища даних [2]. Одна з гілок таких баз даних – графові, де дані зберігаються не у вигляді таблиці, а у вигляді графа. Основними елементами графової моделі є вузли і зв'язки [3].

Графові бази даних мають перевагу при роботі з даними, для яких саме зв’язки між об’єктами відіграють важливу роль, особливо якщо є необхідність відстежувати зв’язки на декілька рівнів вглибину. У той час як реляційні бази даних обчислюють з’єднання під час запиту через важкі операції join, графова база даних зберігає з'єднання разом з даними в моделі [4].

Зазначені преваги та сучасні тенденції з вирішення деяких задач на основі збереження інформації у вигляді графів та питання оптимальної обробки самих відношень та зв’язків між ними, спонукали обрати графову модель збереження даних при розробці експертної системи, для якої природною є графова структура даних. Вибір саме графової БД для реалізації семантичної мережі в ЕС може спростити складність алгоритмів саме завдяки своїй архітектурі.

Аналіз останніх досліджень і публікацій

У зв'язку із бурхливим ростом обсягів інформації на початку 2000-х розвиток отримав напрямок NoSQL. Зокрема, огляду, аналізу та порівнянню реляційних та нереляційних баз даних присвячено ряд публікацій [5-11]. Але наведені дослідження не стосувалися проектування та розробки експертних систем з використанням семантичних мереж у якості моделі представлення знань.

Формулювання мети дослідження

Метою роботи було порівняння ефективності застосування реляційної та нереляційної моделей збереження даних для побудови бази знань експертної системи з використанням семантичної мережі. На прикладах формування одних і тих самих запитів з використанням різних моделей даних, з'ясувати, чи є графова модель більш доцільним рішенням при проектуванні та створенні семантичних мереж, де потрібно зберігати та обробляти складні ієрархічні зв'язки між об'єктами.

Викладення основного матеріалу дослідження

Серед відомих графових БД можна виділити AllegroGraph; FlockDB; Giraph; Apache HyperGraphDB; InfoGrid; Neo4j; SparkSee, Titan; [12].

Neo4j – найбільш популярна графова БД, побудована на основі сітьової моделі даних з відкритим вихідним кодом, реалізована на Java. Станом на 2015 рік вважається найпоширенішою графовою БД.

В графовій БД Neo4j використовується власна мова запитів та маніпулювання даними – Cypher. Але запити можна робити також в інший спосіб, наприклад, безпосередньо через Java API і на мові Gremlin, створеному в проекті з відкритим вихідним кодом TinkerPop. Інтерфейс програмування додатків для СУБД реалізований для багатьох мов програмування, включаючи Java, Python, Clojure, Ruby, PHP, також реалізовано API в стилі REST. Neo4j має обмеження: не більше 34 мільярдів вузлів.

Архітектура Neo4j розроблена для оптимізації таких процесів, як управління, зберігання та обхід вузлів і зв'язків. В Neo4j відношення є важливими об'єктами, які представляють собою зв'язки між сутностями. Операція, що зазначена вище та відома в реляційній базі як об'єднання (join), продуктивність якої падає експоненційно з числом відносин, в Neo4j здійснюється як навігація від одного вузла до іншого, алгоритм роботи якого є лінійним [13].

Версії СУБД, які використовувались для тестування: Neo4j 3.2.3, MySQL 5.6(x64).

Тестування відбувалося на персональному компютері з наступними характеристиками: процесор Intel(R) Core(TM) i5-3210M CPU @2.50GHz, 64-розрядна операційна система, процесор x64, обсяг оперативної пам'яті 6 ГБ.

Для проведення порівняння необхідно насамперед створити таблиці у MySQL, а далі заповнити бази MySQL та Neo4j однаковими даними.

В MySQL задано наступну структуру таблиці test_table, в якій передбачено лише два атрибути: out_node – значення батьківського вузла;

```
CREATE TABLE `test_table` (
  `out_node` int(11) NOT NULL,
  `in_node` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
ALTER TABLE `test_table`
  ADD UNIQUE KEY `out` (`out_node`),
  ADD UNIQUE KEY `in` (`in_node`);
```

На обидва поля встановлено BTREE індекси для прискорення операції зчитування даних. Було створено 1000 записів. Нульовий елемент (колонка out_node) зв'язаний з першим (колонка in_node), перший (колонка out_node) з другим (колонка in_node) і таким чином до останнього.

В Neo4j задано наступну структуру полей:

```
CREATE (test[Index] {value: [Index] })-[:LINK]->(test[Index] +1 {value: [Index] +1})
```

Індекси на вузлах не застосовувались. Для виміру використовувалась вся потужність заданої множини (1000 елементів).

Для генерації тестових даних написано код на мові PHP:

```
<?php
$mode = $argv[1];
$count = $argv[2] ?: 1000;

if ($mode == 'sql') {
        $sql_code = "INSERT INTO `test_table` (`out_node`, `in_node`) VALUES\n";
        for ($i = 0; $i < $count; $i++) {
                $sql_code .= sprintf('(%d, %d),', $i, $i + 1) . "\n";
        }
        $sql_code = rtrim($sql_code, "\n,") . ';';
```

```
                file_put_contents('generator.sql', $sql_code);
        }
        if ($mode == 'cypher') {
                $cypher_code = 'CREATE ';
                for ($i = 0; $i < $count; $i++) {
                        $cypher_code .= sprintf('(test%s)-[:LINK]->(test%s {value: %s}),', $i, $i + 1, $i + 1) .
"\n";
                }
                $cypher_code = rtrim($cypher_code, "\n,");

                file_put_contents('generator.cypher', $cypher_code);
```

Для порівняння ефективності двох СУБД використані наступні показники: певний набір запитів, швидкість їх обробки, простота розуміння та програмної реалізації запитів. Зокрема, розглянуто два запити: пошук кореневого вузла (Q1) та багаторівнева вибірка вузла (Q2).

Запит Q1 в Neo4j: MATCH (n) WHERE NOT (n)<-[]-() RETURN n

Час виконання запиту Q1 у графовій БД Neo4j дорівнює 3 мс; результат його виконання зображено на рис. 1.

Запит Q1 мовою SQL:

SELECT out_node FROM `test_table` WHERE out_node NOT IN (SELECT in_node FROM `test_table`);

Час виконання запиту Q1 у реляційній БД дорівнює 0,2 мс; результат його виконання зображено на рис. 2.

Запит Q2 в графовій базі даних Neo4j:

MATCH result = ({ value:0 })-[*]->({ value:1000 }) RETURN result

Час виконання запиту Q2 у графовій СУБД Neo4j дорівнює 3 мс; результат його виконання зображено на рис. 3.

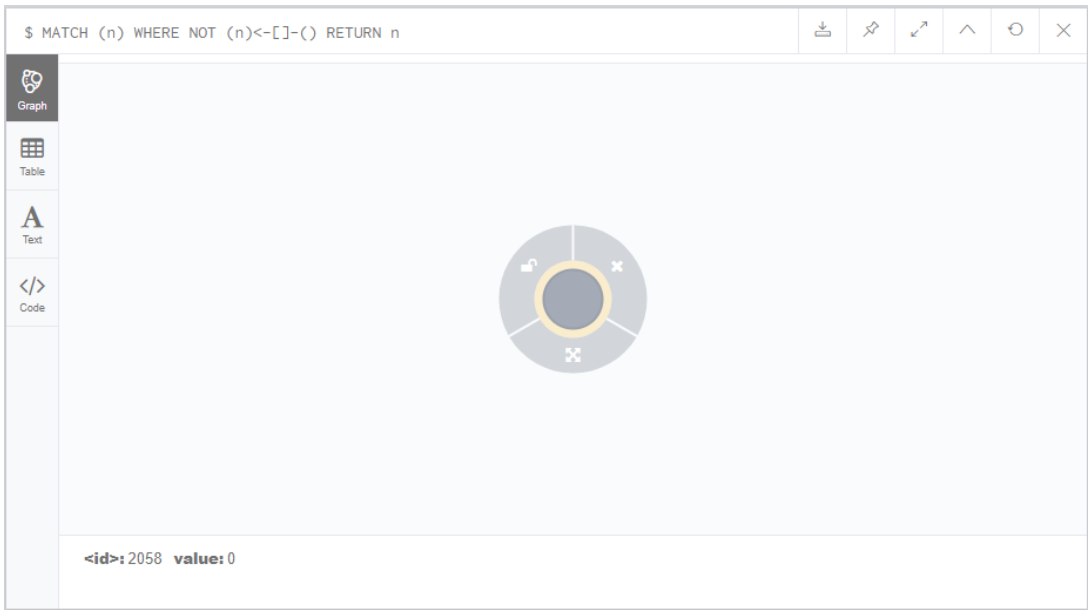

Рис. 1. Результат виконання запиту Q1 у графовій БД Neo4j

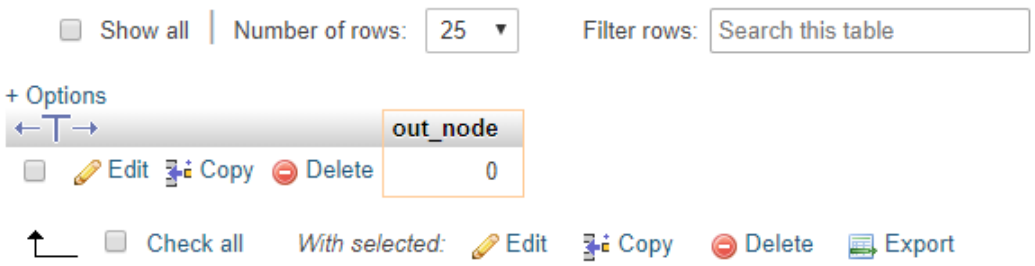

Рис. 2. Результат виконання запиту Q1 у реляційній БД MySQL

Рис. 3. Результат виконання запиту Q2 у графовій БД Neo4j

Запит Q2 мовою SQL: SELECT out_node, in_node FROM (SELECT * FROM test_table ORDER BY out_node, in_node) nodes_sorted,

(SELECT @pv := '1') initialisation

WHERE find_in_set(out_node, @pv) AND length(@pv := concat(@pv, ',', in_node)) AND in_node != 1000;

Час виконання запиту Q2 у реляційній БД Neo4j дорівнює 1.2 мс; результат його виконання зображено на рис. 4.

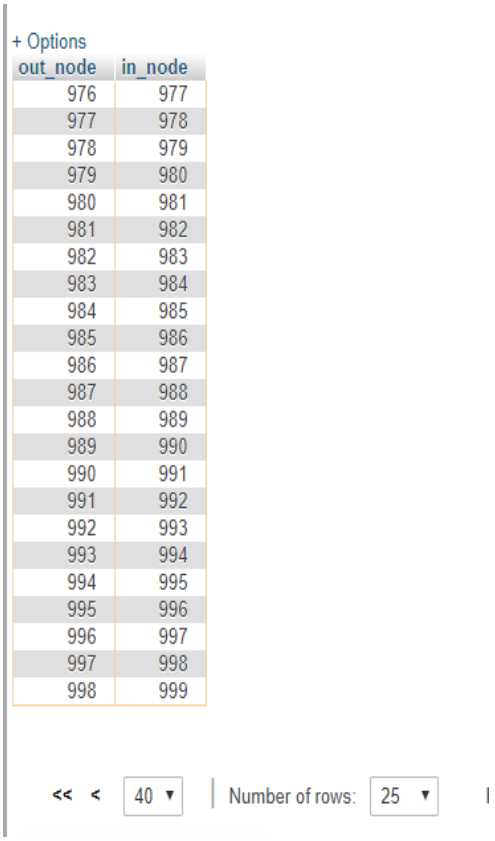

Рис. 4. Результат виконання запиту Q2 у реляційній БД MySQL

Функція find_in_set буде суттєво сповільнюватися із збільшенням записів у таблиці. Також потенційно може спотворитися порядок виконання команд запитів, якщо розташувати запит всередині іншого.

Висновки

1. За результатами тестування визначені час виконання двох типів запитів на обсязі 1000 записів (вузлів) та наведені скрипти для оцінки простоти розуміння програмної реалізації коду.
2. Проведене порівняння на основі розроблених запитів показало, що обидві БД впорались із розглянутими у тестах запитами на заданому обсязі даних однаково ефективно. Час пошуку в реляційній і графовій базі даних відрізняється не суттєво.
3. Продемонстровано, як за рахунок нестандартних алгоритмічних рішень та сторонніх модулів, які розширюють можливості SQL, можна реалізувати роботу із ієрархічною структурою в SQL, як це зроблено при реалізації запиту Q2 (багаторівнева вибірка вузла).
4. Не зважаючи на привабливість графової структури для реалізації семантичної мережі в ЄС, де потрібно зберігати та обробляти складні ієрархічні зв'язки між об'єктами, традиційний підхід з використанням реляційної БД також може бути застосовано, оскільки обсяг даних не достатній для виявлення переваг графової БД, таких як високі показники швидкості виконання запитів та лаконічність коду.

Список використаної літератури

1. Hunger M., Boyd R., Lyon W. The Definitive Guide to Graph Databases for the RDBMS Developer: e-book. 2016. 28 p. URL: https://neo4j.com/whitepapers/rdbms-developers-graph-databases-ebook/?utm_source=odbms&utm_campaign=dl&utm_expid=.Iz8KdWvSRy2NIqH2l3CMPw.0&utm_referrer=http%3A%2F%2Fwww.odbms.org%2F2016%2F02%2Fthe-definitive-guide-to-graph-databases%2F (дата звернення: 11.10.2018).
2. NoSQL базы данных: понимаем суть [Електронний ресурс]. – Режим доступу: http://habrahabr.ru/post/152477/. – Загол. з екрану.
3. Robinson I, Webber J, Eifrem E. Graph Databases. – O'Reilly Media, 2015. – 178 p.
4. Мелешко Е.А. Причины и предусловия применения нереляционных баз данных / Мелешко Е.А., Лозицкая Л.Г. [Електронний ресурс]. – Режим доступу: http://www.rusnauka.com/4_SND_2012/Informatica/4_99281.doc.htm.
5. A comparison of a graph database and a relational database: a data provenance perspective / Chad Vicknair et.al. ACM SE '10 Proceedings of the 48th Annual Southeast Regional Conference, Oxford, Mississippi — April 15 - 17, 2010. NY:ACM New York, 2010. Article No. 42. table of contents ISBN: 978-1-4503-0064-3 doi>10.1145/1900008.1900067.
6. Глибовець А.М. Порівняння Neo4 і реляційної бази даних MySQL / А.М. Глибовець, А.О. Добрянський // Наукові записки НаУКМА. Комп'ютерні науки. – 2015. – Т. 177. – С. 108-112. – Режим доступу: http://nbuv.gov.ua/UJRN/NaUKMAkn_2015_177_21
7. R. Angles, C. Gutierrez Survey of graph database models. ACM Computing Surveys, Vol. 40, No. 1, Article 1. New York, NY: ACM, February 2008. 39 p.
8. Salim Jouili, Valentin Vansteenberghe. An empirical comparison of graph databases. SOCIALCOM '13: Proceedings of the 2013 International Conference on Social Computing, Alexandria, VA, USA, September 08-14, 2013. Washington: IEEE Computer Society, 2013. P. 708-715. doi>10.1109/SocialCom.2013.106.
9. Garima Jaiswal, Arun Prakash Agrawal. Comparative analysis of Relational and Graph databases. IOSR Journal of Engineering (IOSRJEN). Vol. 3, Issue 8 (August. 2013), ||V2|| P. 25-27. URL: http://www.iosrjen.org/Papers/vol3_issue8%20(part-2)/E03822527.pdf (дата звернення: 11.10.2018). DOI: 10.9790/3021-03822527.
10. Srinath Srinivasa Data, Storage and Index Models for Graph Databases. Graph Data Management: Techniques and Applications, IGI Global, Chapter 3, 2012. P. 47-71. DOI: 10.4018/978-1-61350-053-8.ch003.
11. A Comparison between a Relational Database and a Graph Database in the context of a Personalized Cancer Treatment Application / Martinez A. et.al. Alberto Mendelzon International Workshop on Foundations of Data Management, Panama City, Panamá, June 2016. AMW. Vol.1644. URL: http://ceur-ws.org/Vol-1644/paper37.pdf.
12. List Of NOSQL Databases [Електронний ресурс]. – Режим доступу: http://nosql-database.org/. – Загол. з екрану.
13. Cypher, The Graph Query Language: [Електронний ресурс]. – Режим доступу: https://neo4j.com/cypher-graph-query-language – Загол. з екрану.