УДК 004.652.4

Глибовець А. М., Добрянський А. О.

ПОРІВНЯННЯ NEO4 І РЕЛЯЦІЙНОЇ БАЗИ ДАНИХ MYSQL

У роботі проведено порівняльний аналіз можливостей графової бази даних Neo4 і реляційної бази даних MySQL.

Ключові слова: Neo4j, MySQL, бази даних, NoSQL.

Вступ

Швидкі темпи збільшення обсягу інформації вимагають нових підходів до вирішення проблеми її збереження. Часом вдається модернізувати вже наявні рішення, але інколи потрібні нові засоби, які принципово відмінні від попередників. Більшою мірою мають попит продукти, які використовують реляційну базу даних (РБД), але поряд із ними стрімко розвиваються і NoSQL бази даних. Одна з гілок таких баз даних – графові бази даних [1].

Мета цієї роботи полягає в аналізі графових баз даних, особливостей їх застосування, переваг та недоліків на прикладі порівняння Neo4 і реляційної бази даних MySQL.

В основу РБД покладено реляційну модель даних. Така база даних сприймається користувачем як набір нормалізованих відношень різного ступеня [1], представлених у вигляді взаємопов'язаних таблиць. Відомим недоліком РБД є проблема масштабування. Виникнення та розвиток нереляційних баз даних спрямовані на її вирішення [5].

Нереляційні бази даних використовуються ще з кінця 90-х років, а їх широке використання почалося з 2009 р. [2]. Останні розділяють на декілька типів, залежно від їх масштабування, систем збереження даних, моделей даних та запитів. До єдиної класифікації не дійшли, тому виокремимо основні: ключ/значення, документо-орієнтовані, стовпчиково-орієнтовані та графові.

База даних типу ключ/значення зручна для організації даних. Вона дає змогу зберігати за деяким ключем будь-які дані. У документо-орієнтованих базах кожен запис зберігається як окремий документ, що має власний набір полів. Стовпчиково-орієнтовані бази зберігають дані не у вигляді кортежів, а стовпчиками. Зрештою, для представлення даних у графовій базі даних використовують вершини та ребра, які їх з'єднують [2].

Загальні відомості про Neo4j

Neo4j – open-source графова база даних, історія якої почалася з інвестицій компанії «Neo Technology» в 2003 р. З 2007 р. вона стала публічно доступною. Для Neo4j характерним є дотримання ACID, можливість розбиття на кластери і відновлення після збою в системі. Багато спеціалістів вважають Neo4j лідером серед графових баз даних [6].

ACID (Atomicity, Consistency, Isolation, Durability) – набір властивостей, які гарантують надійну роботу транзакцій. Атомарність (atomicity) передбачає, що кожна транзакція не буде виконана частково. Будуть виконані або всі операції, які входять до неї, або жодна. Узгодженість (consistency) передбачає передумови виконання транзакції. Відповідно до них база даних має перебувати в узгодженому, несуперечливому стані. Ізольованість (isolation) передбачає видимість будь-яких змін за межами транзакції тільки після її повного закінчення. Довговічність (durability) передбачає, що після завершення транзакції дані буде збережено та відновлено в разі збоїв [1].

Компоненти графової бази даних – вузли та ребра. Вони можуть бути доповнені власним набором полів. Графову базу даних схематично зображено на рис. 1.

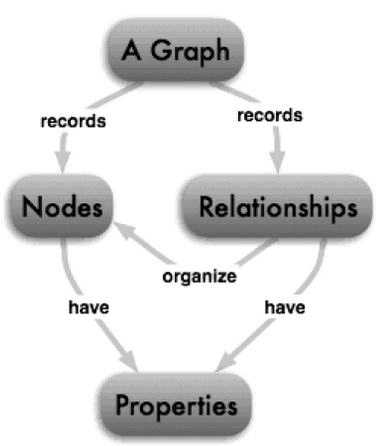

Рис. 1. Структура графової бази даних

Серед особливостей Neo4j виділяють такі: збереження зв'язків під час їх створення, що дає змогу виконувати запити без затримки часу виконання; константний час пошуку зв'язків у графі між вузлами як у ширину, так і в глибину; рівноправність усіх відношень; компактне збереження графів, що дозволяє ефективно масштабувати базу даних; реалізацію на JVM [3].

Підготовка до порівняння

Для проведення порівняння потрібно обрати предметну область, розробити ER-модель, конвертувати ER-модель у реляційну, підняти відповідні бази даних, створити потрібні таблиці в MySQL, заповнити Neo4j та MySQL однаковими даними, отримати статистичні дані.

Предметною областю для дослідження ми обрали соціальну мережу, яка містить користувачів, їхні повідомлення, групи, аудіо та фото.

Розроблену ER-модель зображено на рис. 2.

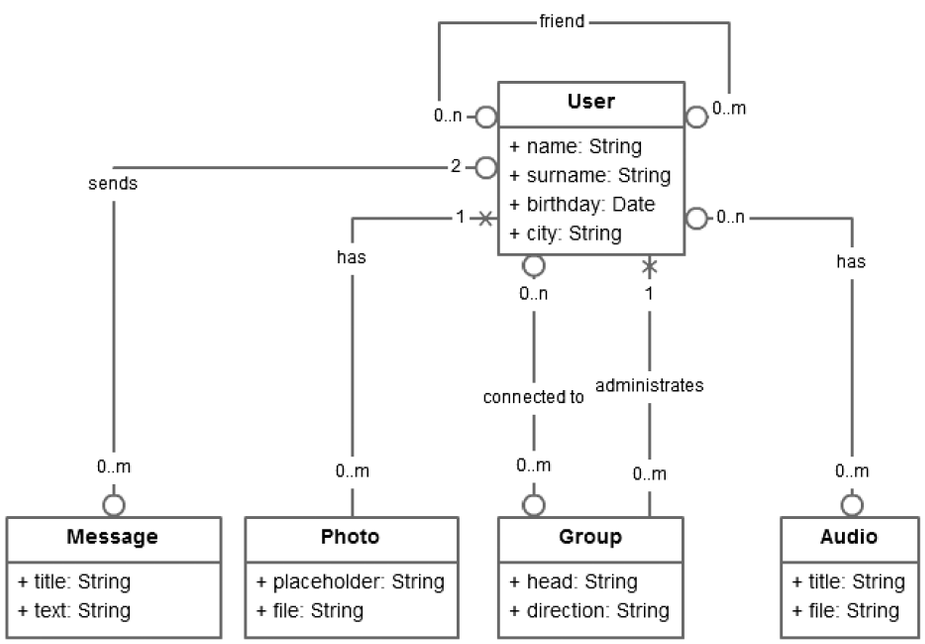

Рис. 2. ER-модель

Для створення ER діаграми було використано online сервіс creately.com. При перетворенні цієї ER діаграми в реляційну додалися нові таблиці: user_audio, user_group, user_photo та friendship.

Для того щоб користуватися Neo4j та MySQL, потрібно:

- встановити MySQL Installer (https://dev.mysql.com/downloads/installer/) для доступу до програмних продуктів, які використовуються для роботи з MySQL;
- завантажити та встановити сервер для роботи з (http://neo4j.com/download/).

За замовчуванням MySQL сервер піднімається автоматично. Для редагування та моніторингу стану цієї бази даних було використано програмний продукт MySQL Workbranch.

Для роботи з реляційною базою даних ми створили такі сутності: Audio, Friendship, Group, Message, Photo, User, UserAudio, UserGroup, UserPhoto.

Заповнення баз даних екземплярами цих сутностей відбувалося у два етапи: генерування даних та вставка в MySQL, копіювання даних з MySQL у Neo4j.

Для генерації тестових даних було створено два класи: Creator (відповідає за створення нових сутностей) та NameGenerator (відповідає за створення стрічок, схожих на ім'я).

Ми створили два DAO: MySQLDAO (для роботи з реляційною базою даних) та Neo4j (для роботи з графовою базою даних). Встановлення з'єднання з MySQL проводилося з використанням JDBC driver (Java DataBase Connectivity) – технології, яка дозволяє взаємодію java коду з базою даних, використовуючи URL до бази даних, логін та пароль користувача.

Для встановлення зв'язку з Neo4j достатньо мати необхідні бібліотеки та створену базу даних. Набір цих бібліотек можна знайти в папці з інстальованим Neo4j сервером.

Фрагмент програмного коду 1 зображує підняття графової бази даних, встановлення з нею зв'язку, обрахування затраченого часу та виведення його на екран.

Фрагмент програмного коду 1. **Частина класу Neo4jDao**

```
static
{
   System.out.println("Opening graph db");
   long time = System.currentTimeMillis();
   graphDb = new GraphDatabaseFactory()
.newEmbeddedDatabase("C:\\Users\\Andriy\\
Documents\\Neo4j\\default.graphdb");
   registerShutdownHook(graphDb);
   System.out.println("Time spent = " + String.
valueOf((-time + System.currentTimeMillis())/
1000));
}
```

Реалізація методів у DAO принципово відмінна. Для роботи з реляційною базою даних використовуються PreparedStatement, створені на основі SQL запитів. Роботу з Neo4j забезпечували бібліотечні методи.

Для об'єктивності аналізу потрібні великі обсяги даних. На невеликій вибірці різниця може бути не суттєва. Тому необхідно згенерувати дані для MySQL не менше ніж на 300 мегабайт. Для цього було створено клас MySQLDatabaseInfilling.

Дані з MySQL копіювалися за допомогою розробленого класу Neo4jDatabaseInfilling. Враховуючи, що кожна таблиця в MySQL мала властивість auto increment, ми можемо отримувати значення з бази даних за ідентифікатором.

Після заповнення обох баз даних було виявлено, що місце, потрібне для того, щоб зберігати одні й ті самі дані, суттєво відрізняється: для MySQL необхідно 351,5 МБ, а для Neo4j – 3,45 ГБ.

Після заповнення баз даних потрібно вибрати критерії для порівняння. Ми зробили наголос на ефективності пошуку в колекції та складності написання запитів.

Аналіз порівнянь

Ми проводили аналіз на основі трьох дослідів.

Дослід 1. Вимірювання часу пошуку користувача за його ідентифікатором.

Щоб зібрати статистичні дані, потрібен метод, який вимірюватиме затрачений час на пошук у реляційній і нереляційній базі даних. Фрагмент програмного коду 2 зображає реалізацію цього методу, який був використаний для цього досліду, а фрагмент програмного коду 3 ілюструє його виклик.

Фрагмент програмного коду 2. **Метод отримання статистичних результатів для досліду**

```
static void findUserByIDTest(int count){
   System.out.print("_____");
   System.out.println("STATISTIC FOR USER ID\
nCOUNT: " + count);

   {
      Random r = new Random(count);
      long time = – System.currentTimeMillis();

      for (int i = 0; i < count; i++) {
         int id = r.nextInt(100000);
         User u = MySQLDAO.getUserByID(id);
      }
      time += System.currentTimeMillis();
      System.out.println("Time for MySQL:" + time);
   }
   {
      Random r = new Random(count);
      long time = – System.currentTimeMillis();
      for (int i = 0; i < count; i++) {
         int id = r.nextInt(100000) + 10;
         User u = Neo4jDAO.getUserByID(id);
      }
      time += System.currentTimeMillis();
      System.out.println("Time for Neo4j:" + time);
   }
   System.out.print("_____");
}
```

Фрагмент програмного коду 3. **Отримання статистичних результатів для досліду**

```
public static void main(String[] args){
   // Database connection initialisation
   init();
   int [] counts = {10, 100, 1000, 5000, 10000, 30000,
60000, 90000, 120000};
   for(int i = 0; i < counts.length; i++){
      findUserByIDTest(counts[i]);
   }
}
```

Дані цього експерименту подано в табл. 1. Діаграму, побудовану на основі результатів проведеного досліду, зображено на рис. 3.

Таблиця 1. **Залежність часу від кількості запитів**

User count = 100000		
Number of queries	**Time for MySQL, ms**	**Time for Neo4j, ms**
10	4	9
100	34	76
1000	286	510
5000	1034	1103
10000	1340	1187
30000	3390	1384
60000	6876	2102
90000	10537	3175
120000	14033	3858

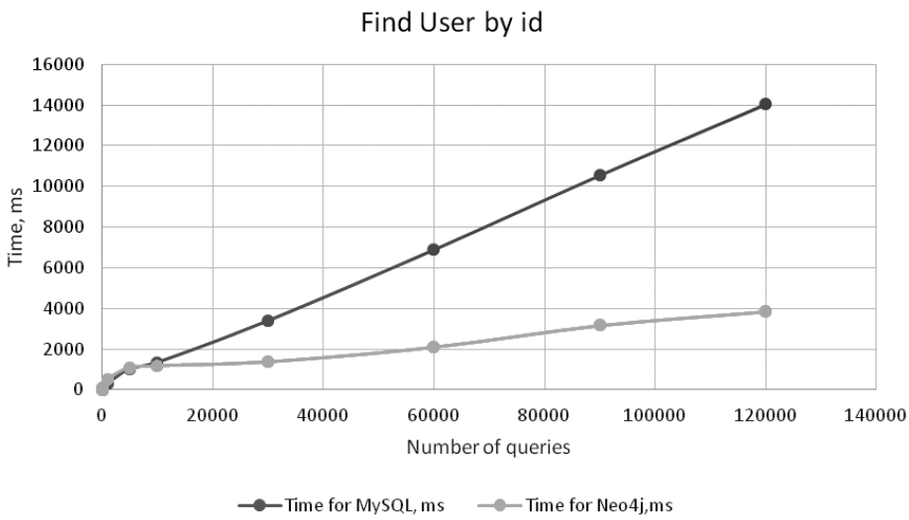

Рис. 3. Діаграма залежності часу від кількості запитів

Дослідивши діаграму, зображену на рис. 3, можна зробити висновок, що пошук за індексованим полем на великій кількості даних у Neo4j набагато швидший, ніж у MySQL (більше ніж у 3 рази).

Дослід 2. Вимірювання часу знаходження друзів користувача зі зміною величини інтервалу входження ідентифікаторів для пошуку.

Для отримання статистичних даних було використано аналогічний метод, як і в досліді 1.

Табл. 2 демонструє результати пошуку друзів користувача за ідентифікатором з проміжку від 0 до 5000, а на рис. 4 представлено ці результати у вигляді діаграми.

Таблиця 2. **Залежність часу від кількості запитів на вибірці до 5000**

Friendship count = 1000000, random range = 5000		
Number of queries	**Time for MySQL, ms**	**Time for Neo4j, ms**
10	4	68
100	37	367
1000	249	975
5000	1025	1521
10000	1478	723
30000	4082	1008
60000	8132	1248
90000	12065	1848
120000	15622	2433
200000	26639	4043
300000	39758	6049
400000	52290	8037
500000	68743	10148

Аналогічним чином було отримано дані для проміжків від 0 до 250000 та від 0 до 500000. На основі них було побудовано діаграму, яка зображена на рис. 5.

Проаналізувавши діаграму, зображену на рис. 5, бачимо, що час пошуку в Neo4j на великій кількості даних кращий, ніж у MySQL. Також на одних і тих самих даних пошук швидший, якщо робити вибірку з невеликого діапазону. Це можна пояснити роботою оптимізаторів. Збільшення діапазону не завжди спричиняє збільшення часу пошуку.

Висновки

Дані в графовій базі даних займають більше місця, ніж у реляційній. Час на оптимізацію графової бази даних більший, ніж у реляційної. Користувач графової бази даних, розробляючи клієнтський код мовою Java, може не знати мови запитів Cypher для того, щоб виконувати елементарні операції з Neo4j. Синтаксис Cypher зрозуміліший і займає менший обсяг, ніж SQL запит. Час пошуку в реляційній і графовій базі даних на невеликих обсягах даних відрізняється не суттєво, здебільшого MySQL показувала кращі результати, ніж Neo4j. Графова база даних суттєво переважає реляційну в часі пошуку на великих обсягах даних.

Відповідно до отриманих статистичних даних, графова база призначена для швидкого пошуку у великій колекції, яка може мати гнучку структуру даних. Цю гнучкість реляційна база не підтримує, її призначення – зберігати дані з чіткою визначеною структурою.

Отже, графову базу даних слід використовувати, коли є великі обсяги даних і ресурси жорсткого диску не так важливі, а пошук повинен здійснюватися швидко. Для невеликих систем, які мають чітку незмінну структуру даних, краще використовувати реляційну базу даних.

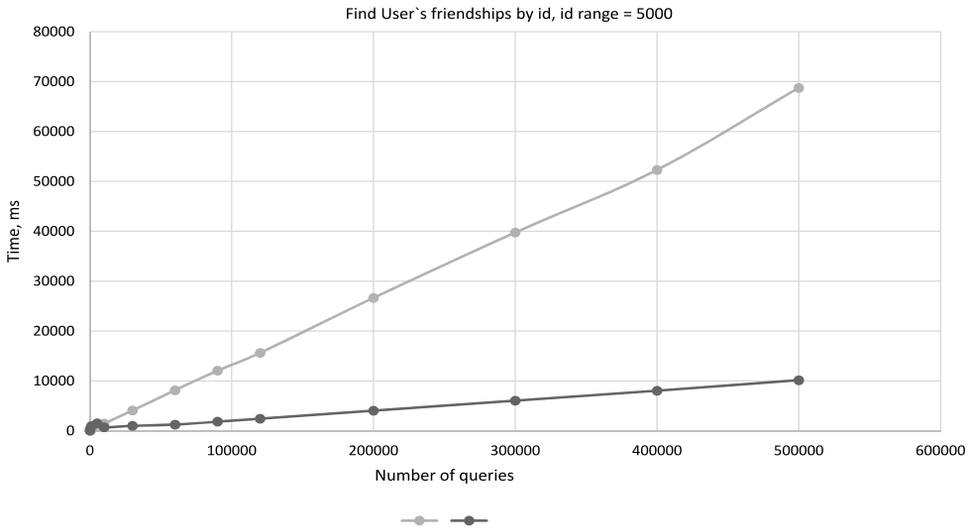

Рис. 4. Діаграма залежності часу від кількості запитів на вибірці до 5000

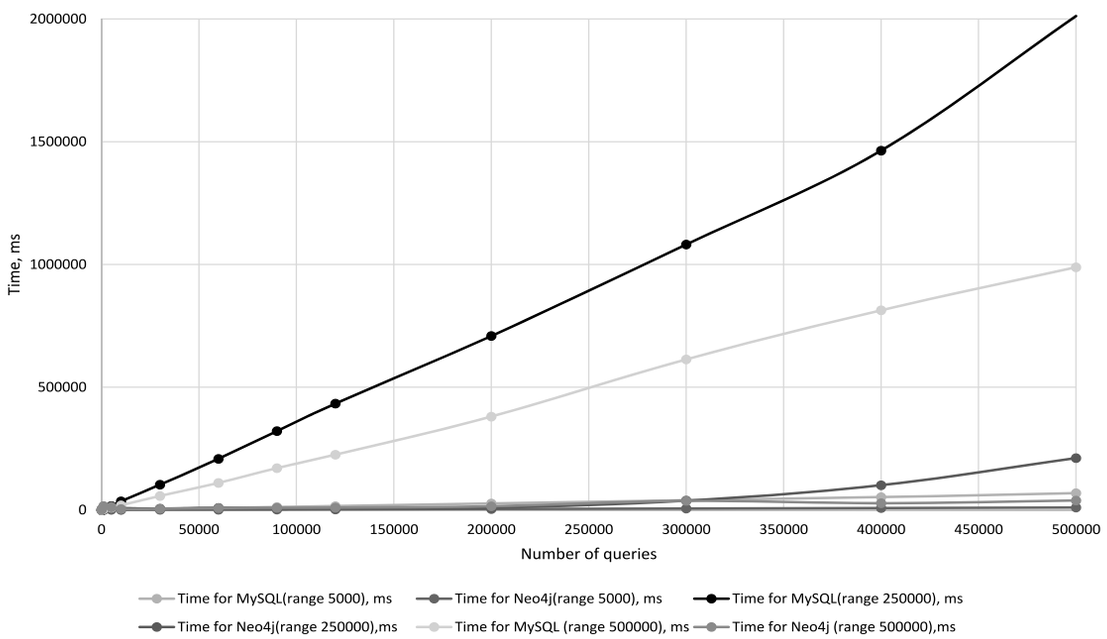

Рис. 5. Діаграма залежності часу від кількості запитів на декількох вибірках

Список літератури

1. Глибовець М. М. Інтелектуальні мережі : навч. посіб. / М. М. Глибовець, А. М. Глибовець, М. В. Поляков. – Дніпропетровськ : Нова ідеологія, 2014. – 464 с.
2. Причины и предусловия применения нереляционных баз данных [Электронный ресурс]. – Режим доступа: http://www.rusnauka.com/4_SND_2012/Informatica/4_99281.doc.htm. – Загл. с экрана.
3. Graph modelization of Neo4j internal storage [Electronic resource]. – Mode of access: http://www.neo4j.org/graphgist?76d0043072143a53d789f. – Title from the screen.
4. Neo4j [Electronic resource]. – Mode of access: http://neo4j.com/.– Title from the screen.
5. NoSQL базы данных [Электронный ресурс]. – Режим доступа: http://habrahabr.ru/post/152477/. – Загл. с экрана.

A. Glybovets, A. Dobrianskiy

COMPARISON OF NEO4 WITH RELATIONAL DATABASE MYSQL

In this work the comparative analysis was conducted between graph database Neo4j and relational database MySQL.

Keywords: Neo4j, MySQL, database, NoSQL.

Матеріал надійшов 19.10.2015