А. М. Глибовець[1], K. Haenssgen[2]

Neo4j як ядро рекомендаційної системи

[1] Національний університет «Києво-Могилянська академія», Київ, Україна.

[2] Hochschule fur Technik, Wirtschaft und Kultur Leipzig, Leipzig, Germany

Вступ. На даний момент актуальним є питання побудови рекомендаційних систем, що можуть працювати з великими даними[1]. Можливим вирішенням цих проблеми є використання графової СКБД Neo4j в якості ядра рекомендаційної системи. Метою цієї статі є дослідження придатності Neo4j до створення рекомендаційних систем та розробка прототипу рекомендаційної системи.

Масштабування Neo4j. Всі СКБД які хочуть відповідати сучасним вимогам повинні гарно масштабуватися. Neo4j підтримує горизонтальне масштабування master-slave [2]. Кожен вузел розбитий на дві частини, безпосередньо база даних і компонент управління кластером. Цей компонент постійно тримає зв'язок із іншими вузлам, і приймає відповідні рішення. Коли необхідно обрати головний вузол, цей компонент перевіряє чи правильно був обраний головний вузол. Це відбувається у автоматичному режимі, що є зручним для розробників оскільки не вимагає ручної конфігурації. Не головні вузли підтягують транзакційні оновлення з головного вузла, отже всі операції запису узгоджуються із головним вузлом. Це є недоліком, тому що необхідно оновлювати дані в головному вузлі, або чекати на синхронізацію вузлів. Пріоритетним є перший варіант оскільки він є швидшим. Ще одним недоліком кластерної архітектури Neo4j є те, що весь граф повністю копіюється на підконтрольні вузли. З однієї сторони це добре оскільки при відмові всіх вузлів окрім одного залишається вузол з останніми правильним даними. Але це надмірне використання ресурсів, що може проявитися при збережені великих даних. Один вузол Neo4j може зберігати 34 мільярди вершин, 34 мільярдів зв'язків а також 68 мільярдів атрибутів, лише компанії як Google, Facebook можуть перевищити цей ліміт, тож поки не варто турбуватись через це.

На відміну від популярних СКБД, що дозволяють розподіляти дані по декількох серверах, в Neo4j такої можливості досі не реалізовано.Як відмічає автор у [3], математична проблема

оптимального розбиття великого графу між декількома серверами є NP-повною задачею, тобто її дуже важко вирішити швидко і ефективно. Дана проблема є відкритою.

Реалізація рекомендаційної системи художніх книжок. В ході написання статті було вирішено реалізувати рекомендаційну систему художніх книжок. Для генерації тестових даних було використано колекцію книжок доступних в Інтернет (http://lib.ru). Для генерації оцінок користувачів було розроблено алгоритм який випадковим чином генерує оцінки від одного до п'яти, подобається/не подобається.

Процес рекомендації складається з чотирьох кроків:

1. Знайти всіх користувачів що оцінили хоча б одну книжку, назвемо цих користувачів сусідами.
2. Для кожного сусіда знайти книжки які він оцінив, і оцінив користувач для якого робиться рекомендація.
3. Для множини книжок які оцінили обидва користувачі обчислити евклідову відстань між двома користувачами.
4. Створити зв'язок між сусідами і користувачем для якого робиться рекомендація.

Реалізація алгоритму на мові Cypher

```
MATCH (u1:User {name: 'Anna Bowman'})-[x:LIKES]->(b:Book)<-
[y:LIKES]-(u2:User) WITH SUM(x.rating * y.rating) AS xyDotProduct,
SQRT(REDUCE(xDot = 0.0, a IN COLLECT(x.rating) | xDot +
a^2)) AS xLength, SQRT(REDUCE(yDot = 0.0, b IN COLLECT(y.rating)
| yDot + b^2)) AS yLength,u1, u2 MERGE (u1)-[s:Similarity]-
>(u2) SET s.similarity = xyDotProduct / (xLength * yLength)
MATCH (u1:User)-[r:LIKES]->(b:Book),
(u1)-[s:Similarity]-(u2:User {name:"Andrii Glybovets'})
WHERE NOT((u2)-[:LIKES]->(b))
WITH b, s.similarity AS similarity,
r.rating AS rating
ORDER BY b.name, similarity DESC
WITH b AS book, COLLECT(rating)[0..3] AS ratings
WITH book, REDUCE(s = 0, i IN ratings | s + i)*1.0 / LENGTH(ratings)
AS reco ORDER BY reco DESC
RETURN book AS Book, reco AS Recommendation
```

Neo4j дозволяє робити більш складні рекомендації, наприклад рекомендації базовані на друзях користувача, запит наведено нижче.

Рекомендація з використання друзів користувача

MATCH (***user***:***User***)-[:IS_FRIEND_OF]->(friend), (friend)-[:LIKES]->(book:Book),
WHERE ***user***.**name** = **'Andrii Glybovets'**
RETURN book.**title**, count(*) AS occurrence ORDER BY occurrence DESC LIMIT 5

Порівняння з реалізацією в процедурному стилі

Основними показниками для порівняння є час необхідний для виконання алгоритму, оперативна пам'ять, точність, повнота. Крім реалізації з використанням Neo4j реалізовано традиційний алгоритми колаборативної фільтрації та Розклад невід'ємних матриць (NMF). Тестування проводилося на однакових даних.

Таблиця 1. Порівняння різних алгоритмів

	100 користувачів	1000 користувачів	5000 користувачів
Колаборативна фільтрація	4,1 сек.	17,2 сек.	61.2 сек.
NMF	4,7 сек.	33,5 сек.	120.6 сек.
Neo4j	5.3 сек.	30.1 сек.	55.2 сек.

Хоча метод розкладу невід'ємних матриць працює найповільніше варто зазначити що він шукає рекомендації для всіх користувачів. Також варто зазначити, що в варіанті Neo4j дані отримуються за один запит, а в двох інших необхідно ще дістати дані з бази даних та обробити їх.

Висновок. В роботі було розглянуто створення рекомендаційної системи за допомогою Neo4j, а також розглянуто придатність Neo4j до сучасних вимог таких як масштабованість та розподіленість даних. Суттєвим недоліком є те що при використанні горизонтально масштабування доводиться копіювати повністю граф. Окрім того було проведено порівняння з традиційними підходами до розробки рекомендаційних систем. В ході тестування було виявлено що рекомендації за допомогою Neo4j є найшвидшими при збільшені кількості користувачів.

[1] Dietmar J. Recommender Systems An Introduction / Jannach Dietmar. – Cambridge: Cambridge university press, 2011. – 353 с.

[2] Rik Van Bruggen. Learning Neo4j / Rik Van Bruggen. – Birmingham: Packt Publishing Ltd., 2014. – 222 с.

[3] David M. Understanding Neo4j Scalability [Електронний ресурс] / David Montag– Режим доступу до ресурсу: http://info.neo4j.com/rs/neotechnology/images/Understanding%20Neo4j%20Scalability

E-mail: ✉[1] *andriy@glybovets.com.ua,* ✉[2] *haenssge@imn.htwk-leipzig.de.*