Методи та структури даних для реалізації бази даних рекомендаційної системи соціальної мережі

Автори публiкації: Міхав В.В., Мелешко Є.В., Шимко С.В. Центральноукраїнський національний технічний університет,

Анотація

Метою даної роботи є дослідження та програмна реалізація методів і структур даних для побудови бази даних рекомендаційної системи, щоб порівняти ефективність їх використання за затратами часу та пам’яті. Наявність великої кількості різних методів реалізації баз даних викликає необхідність порівняльного аналізу та вибору оптимального методу і структури даних для зберігання інформації у рекомендаційних системах.

Було проведено дослідження різних структур даних, які можна використати для створення бази даних рекомендаційної системи, зокрема, досліджені зв’язний список, розгорнутий зв’язний список, хеш-таблиця, B-дерево, B+-дерево та бінарна діаграма рішень. Також було проведено серію експериментів на програмній імітаційній моделі рекомендаційної системи з різною кількістю агентів, предметів та сесій.

Відповідно до результатів проведених експериментів, розгорнутий список показав найкращі показники швидкодії та використання пам’яті. Структура B+-дерево показала результати, близькі до хеш-таблиці. Час доступу до окремого елементу в обох випадках сталий, але B+- дерево має певні переваги – елементи зберігаються відсортованими, а при зміні розміру немає необхідності розширювати область пам’яті. Найгірші результати показала структура даних бінарна діаграма рішень як за затратами часу, так і за затратами пам’яті. Профілювання показало, що 75% часу роботи тесту варіанту з розгорнутим списком зайняло генерування випадкових даних для програмного імітаційного моделювання агентів та предметів рекомендаційної системи, тож, саме сховище даних має високі показники ефективності.

Профілювання варіанту із інвертованим списком показало, що доступ до випадкових блоків займає більше часу через неможливість закешувати їх, тож, за умов реального навантаження час вставки нових даних буде більшим, а відносна ефективність застосування інвертованого списку зросте. Для найбільш ефективного використання пам’яті розмір блоку зв’язного списку має бути адаптований таким чином, щоб блоки були максимально заповнені. Блоки малого розміру зменшують втрати пам’яті, але збільшують час обходу всіх елементів списку та збільшують накладні витрати пам’яті.

Ключові слова: рекомендаційні системи, бази даних, структури даних, програмна імітаційна модель.

Стаття

Постановка проблеми. Рекомендаційні системи є важливою складовою соціальних мереж та значним чином впливають на те, яким користувачі сприймають інформаційний простір [1, 2]. Вибір методу представлення даних, якими оперує рекомендаційна система, має важливе значення, оскільки фективний спосіб побудови бази даних для роботи такої системи може зменшити кількість потрібних ресурсів та збільшити кількість доступних алгоритмів для формування списків рекомендацій. Отже, вибір методів реалізації СУБД для зберігання даних рекомендаційної системи є важливою науково-практичною задачею.

Аналіз останніх досліджень і публікацій. У наш час існує багато різних систем управління базами даних, крім реляційних баз даних широке застосування отримують бази даних типу NoSQL [3, 4]. СУБД типу NoSQL можуть бути реалізовані різними методами, зокрема, як Сховища типу «ключ-значення» (Key-value stores), Масштабовані розподілені сховища (Column Family (Bigtable) stores), графові СУБД (Graph Stores), документо-орієнтовані СУБД (Document Stores) тощо [3-5].

Спосіб зберігання даних рекомендаційної системи є важливим з точки зору якості її роботи, швидкості, можливостей масштабування, зручності виконання основних операцій з даними для формування рекомендацій.

Все частіше для зберігання даних рекомендаційних систем та інших додатків починають використовувати графові моделі [6-8], також графова форма представлення даних стає поширеною у програмному моделюванні складних систем та мереж [9-12], і це відбувається через ряд переваг графових моделей [8, 13]. Яскравим прикладом такого підходу являється побудова рекомендаційних систем з застосуванням графової СУБД Neo4j [14]. Графові моделі СУБД надають не лише зручний формат зберігання даних, а й зручний формат запитів. В документації до Neo4j є приклади реалізації алгоритмів формування рекомендацій запитами до цієї СУБД, що ілюструє її придатність для використання в рекомендаційних системах.

Наявність такої великої кількості різних методів реалізації баз даних та представлення інформації, що можна використати при побудові рекомендаційних систем, викликає необхідність порівняльного аналізу та вибору оптимального методу і структури даних для зберігання інформації рекомендаційних систем.

Постановка завдання. Метою даної роботи є дослідження та програмна реалізація методів і структур даних для реалізації бази даних рекомендаційної системи, щоб порівняти ефективність їх використання за затратами часу та пам’яті.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань: – Дослідження існуючих структур даних для зберігання інформації та методів їх реалізації. – Програмна реалізація досліджених структур даних для створення бази даних рекомендаційної системи. – Проведення серії експериментів для порівняння ефективності використання розглянутих структур даних за затратами часу та пам’яті.

Об’єктом дослідження є процес зберігання даних рекомендаційної системи. Предметом дослідження є методи та структури даних для реалізації бази даних рекомендаційної системи. Методи дослідження базуються на методах розробки програмного забезпечення, теорії побудови баз даних, теорії алгоритмів та теорії статистичної обробки даних.

Виклад основного матеріалу. В статті проведено дослідження різних структур даних, які можна використати для створення бази даних рекомендаційної системи [15-17]. Наведемо класифікацію структур даних, що були обрані для дослідження та підходять для рішення поставленого завдання.

Зв’язний список (linked list) – це структура даних, у якій кожен елемент має вказівник на наступний елемент. Основна перевага цієї структури полягає у сталому часі додавання нового елементу. Проте для кожного елементу потрібно виділяти новий блок пам’яті, через що менеджер пам’яті спричиняє значні затримки та накладні витрати пам’яті.

Розгорнутий зв’язний список (unrolled list) – це зв’язний список, кожен елемент якого містить масив логічних елементів. Це дозволяє об’єднати переваги масивів та зв’язних списків у випадку додавання елементів у кінець списку. Об’єднання блоків логічних елементів у список дозволяє додавати нові елементи без зміни розміру блоку пам’яті, економити пам’ять на вказівниках та ефективніше використовувати кеш процесора завдяки послідовному розташуванню елементів. При послідовному заповненні гарантується, що незаповненим лишиться не більше одного блоку елементів.

Хеш-таблиця (hash map) – це структура даних, у якій пошук елементу здійснюється на основі його ключа. Хеш від ключа вказує, у якій комірці розташовується елемент. Якщо кілька елементів мають однаковий хеш, то виникає колізія. Існує два методи розв’язання колізій – закрита та відкрита адресації. При закритій адресації кожен елемент таблиці – це зв’язний список і усі елементи з однаковим хешем додаються до одного списку. Це найпростіший спосіб розв’язання колізій, але він використовує додаткову пам’ять для вказівників і не дозволяє використовувати переваги кешування при обході елементів хеш-таблиці. При відкритій адресації у випадку колізії обирається нова позиція елементу. Нова позиція може обиратися як за допомогою додаткової хеш-функції, так і шляхом зміщення позиції на декілька елементів. Пошук повторюється, доки не буде досягнуто порожнього елемента. Відкрита адресація використовує фіксований об’єм пам’яті і не потребує додаткових вказівників, але для ефективності операцій вставки і пошуку таблиця має бути заповнена не більш ніж на 50%, тож це спричиняє додаткові витрати пам’яті.

B-дерево (b-tree) – це структура даних представлена збалансованим, сильно розгалудженим деревом пошуку. Кожен вузол В-дерева, крім листків, є упорядкованим списком, у якому чергуються ключі і вказівники на потомків. Ключі вузла вказують інтервал, у якому знаходяться ключі потомку. В+-дерево (B+-tree) відрізняється тим, що воно зберігає усі значення у листкових вузлах, а листкові вузли мають посилання на сусіда, завдяки чому можна обійти усі значення без обходу всього дерева. Завдяки великій розгалуженості дерева підтримується мала висота дерева, що дозволяє переглядати невеликий об’єм даних за один прохід, а завдяки правилам побудови значення зберігаються у порядку зростання ключа.

Бінарні діаграми рішень (BDD) – це економна форма представлення булевих функцій у вигляді орієнтованого ациклічного графу. Вершини графу представляють аргументи функції, листки – її двійкові значення. Для додавання і вилучення ребер та зміни ваги ребер необхідно мати можливість редагувати дані графу. БДР дають можливість зберігати дані у стисненому вигляді та швидко отримувати значення функції за її параметрами, але редагування БДР вимагає складних обчислень. При представленні булевих функцій у формі БДР стало можливим розв’язувати багато проблем, які при традиційних представленнях структур нерозв’язні через значну розмірність таких представлень і складність операцій над ними. БДР можуть успішно застосовуватися фактично в кожній галузі, де потрібно обробляти дискретні структури даних.

Було проведено серію експериментів для порівняння ефективності використання розглянутих структур даних за затратами часу та пам’яті. Результати експериментів наведені у таблицях 1-2 та на рисунках 1-2.

Експерименти проводилися на комп’ютері з процесором AMD Ryzen 5 3600 та 32 Гб оперативної пам’яті. Для формування рекомендацій було використано колаборативну фільтрацію. З метою моделювання рекомендаційної системи розроблено програмну імітаційну модель, в якій було виділено три основні сутності – агент, сесія та предмет. На цій програмній моделі і проводилися експерименти.

Створена програмна імітаційна модель рекомендаційної системи для проведення експериментів працює за наступним принципом: Крок 1. Ініціалізація рекомендаційної системи: задається кількість агентів, предметів та сесій, а також розмір сесії та максимальна кількість вподобань.

Крок 2. Для кожного агента випадковим чином генерується від 1 до n вподобань.

Крок 3. Створюється m сесій. До кожної сесії закріплюється випадковим чином обраний агент. Потім серед вподобань цього агента випадковим чином обирається від 1 до k вподобань, які копіюються до сесії.

Крок 4. Випадковим чином обирається контрольна сесія, для якої буде сформовано рекомендацію.

Крок 5. Визначаються усі предмети, які належать до контрольної сесії.

Крок 6. Здійснюється пошук усіх сесій, вподобання яких мають перетин із вподобаннями контрольної сесії. На цьому етапі є можливість відфільтрувати сесії за розміром перетину.

Крок 7. Визначаються предмети, які буде рекомендовано. Здійснюється пошук усіх предметів, які належать хоча б одній з відібраних сесій, але не належать до контрольної сесії. На цьому етапі є можливість відфільтрувати предмети за кількістю закріплених сесій.

Було проведено 4 серії експериментів. Нижче наведено параметри кожної серії.

Параметри 1 серії експериментів: кількість агентів 65536, кількість предметів 131072, кількість сесій 262144, розмір сесії 192, максимальна кількість вподобань 1536.

Параметри 2 серії експериментів: кількість агентів 131072, кількість предметів 262144, кількість сесій 524288, розмір сесії 256, максимальна кількість вподобань 2048.

Параметри 3 серії експериментів: кількість агентів 262144, кількість предметів 524288, кількість сесій 1048576, розмір сесії 256, максимальна кількість вподобань 2048.

Параметри 4 серії експериментів: кількість агентів 524288, кількість предметів 1048576, кількість сесій 2097152, розмір сесії 256, максимальна кількість вподобань 2048.

У таблиці 1 наведено результати серії експериментів, проведених для порівняння часу формування рекомендацій системою при використанні різних структур даних для реалізації бази даних.

На основі одержаних результатів з таблиці 1 було побудовано графік, наведений на рисунку 1. Як видно з таблиці та рисунку найкращі результати по часу формування рекомендацій показали наступні структури даних – звичайний розгорнутий список та інвертований розгорнутий список. Також непогані результати показала структура даних B+-дерево.

У таблиці 2 наведено результати серії експериментів, проведених для порівняння кількості використаної пам’яті для формування рекомендацій системою при використанні різних структур даних для реалізації бази даних.

На основі одержаних результатів з таблиці 2 було побудовано графік, наведений на рисунку 2. Як видно з таблиці та рисунку найкращі результати за використаною пам’яттю для формування рекомендацій аналогічно, як і з результатами за затратами часу, показали наступні структури даних – звичайний розгорнутий список, інвертований розгорнутий список та B+-дерево.

Профілювання тестового коду показало, що значна відмінність у часі генерації сесій у варіантів з та без інвертованого списку спричинена затримкою доступу до елементів інвертованого списку, оскільки доступ постійно здійснюється до різних елементів. З реальними даними рішення з розгорнутим списком працюватиме дещо повільніше, оскільки кешування буде менш ефективним.

Структура розгорнутого списку дуже проста, тож в подальшому це дасть можливість використовувати багатопотокову роботу без блокувань.

Структура B+ tree показала результати, близькі до хеш-таблиці. Час доступу до окремого елементу в обох випадках сталий, але B+ tree має певні переваги: елементи зберігаються відсортованими, а при зміні розміру немає необхідності розширювати область пам’яті.

Розмір блоку розгорнутого списку впливає на швидкість роботи і на об’єм використаної пам’яті. Зменшення розміру блоку дозволяє зменшити втрати пам’яті, але збільшує час доступу до елементів.

Для прискорення пошуку окремих елементів у розгорнутому списку після заповнення блоку можна відсортувати його елементи. Це дасть можливість використовувати бінарний пошук замість лінійного та перевіряти лише ті блоки, де шуканий елемент належить до інтервалу, утвореного найменшим та найбільшим елементами блоку.

Перевага розгорнутого списку над іншими розглянутими структурами даних у використанні пам’яті значною мірою за рахунок того, що зберігається лише факт вподобання, без параметрів. При використанні 4 байт на елемент для зберігання параметру витрати пам’яті наближаються до максимальних витрат бінарних діаграм рішень.

Висновки. Відповідно до результатів проведених експериментів, розгорнутий список показав найкращі показники швидкодії та використання пам’яті. Профілювання показало, що 75% часу роботи тесту варіанту з розгорнутим списком зайняло генерування випадкових даних для програмного імітаційного моделювання агентів та предметів рекомендаційної системи, тож, саме сховище даних має високі показники ефективності. Профілювання варіанту із інвертованим списком показало, що доступ до випадкових блоків займає більше часу через неможливість закешувати їх, тож, за умов реального навантаження час вставки нових даних буде більшим, а відносна ефективність застосування інвертованого списку зросте. Для найбільш ефективного використання пам’яті розмір блоку зв’язного списку має бути адаптований таким чином, щоб блоки були максимально заповнені. Блоки малого розміру зменшують втрати пам’яті, але збільшують час обходу усіх елементів списку та збільшують накладні витрати пам’яті.

Література

1. Editors Ricci F., Rokach L., Shapira B., Kantor P. B. Recommender Systems Handbook. New York, NY, USA: Springer-Verlag New York, Inc., 2010. 842 p.

2. Valois B.Jr.C., Oliveira M.A. Recommender systems in social networks. JISTEM J.Inf.Syst. Technol. Manag. 2011. Vol.8 No.3. P. 681-716. URL: https://www.scielo.br/scielo.php?script=sci\_arttext&pid=S1807-17752011000300009 (Last accessed: 02.04.2021)

3. Фаулер М., Садаладж П. Дж. NoSQL: Новая методология разработки нереляционных баз данных. М.: Издательский дом «Вильямс». 2013. 192 с.

4. Meier A., Kaufmann M. SQL & NoSQL Databases. Springer Vieweg, Wiesbaden. 2019. P. 201-218. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.7089&rep=rep1&type=pdf (Last accessed: 02.04.2021)

5. Cure O., Blin G. RDF Database Systems: Triples Storage and SPARQL Query Processing. Elsevier Science. 2014. 256 p.

6. Yi N., Li C., Feng X., Shi M. Design and implementation of movie recommender system based on graph database. 14th Web Information Systems and Applications Conference (WISA), IEEE. 2017. P. 132-135.

7. Angles R. A comparison of current graph database models. IEEE 28th International Conference on Data Engineering Workshops, IEEE. 2012. P. 171-177.

8. Засядко Г.Е., Карпов А.В. Проблемы разработки графовых баз данных. Инженерный вестник Дона. 2017. No1(44). URL: https://cyberleninka.ru/article/n/problemy-razrabotki-grafovyh-baz-dannyh (дата обращения: 02.04.2021)

9. Мелков С., Мусатов Д., Савватеев А. Моделирование социальных сетей. 2013. URL: https://kpfu.ru/docs/F117464271/MMS\_socnet\_cities.pdf (дата обращения: 02.04.2021)

10. Берновски М.М., Кузюрин Н.Н. Случайные графы, модели и генераторы безмасштабных графов. Труды ИСП РАН. 2012. URL: https://cyberleninka.ru/article/n/sluchaynye-grafy-modeli-i-generatory-bezmasshtabnyh-grafov (дата обращения: 02.04.2021)

11. Райгородский А.М. Математические модели Интернета. “Квант”. 2012. No4. С. 12-16. URL: https://elementy.ru/nauchno-populyarnaya\_biblioteka/431792 (дата обращения: 02.04.2021)

12. Meleshko Ye. Computer model of virtual social network with recommendation system. Scientific journal Innovative Technologies and Scientific Solutions for Industries, Kharkiv, NURE. 2019. Issue 2(8). P. 80

13. Робинсон Я., Вебер Д., Эифрем Э. Графовые базы данных: новые возможности для работы со связанными данными. М.: ДМК Пресс. 2016. 256 с.

14. Neo4j Documentation. 2020. URL: https://neo4j.com/docs (Last accessed: 02.04.2021)

15. Ахо А.В., Хопкрофт Д., Ульман Д.Д. Структуры данных и алгоритмы. М.: Вильямс, 2000. 384 с.

16. Minato S. Zero-suppressed BDDs and their applications. International Journal on Software Tools for Technology Transfer. 2001. No3.2. pp. 156-170.

17. Кнут Д.Э. Искусство программирования, Том 4А. Комбинаторные алгоритмы. Часть 1. М.: Вильямс. 2013. 960 с.