УДК 004.9 **doi: 10.26906/SUNZ.2021.4.059**

В. В. Міхав, Є. В. Мелешко, М. С. Якименко, Д. В. Бащенко

Центральноукраїнський національний технічний університет, м. Кропивницький, Україна

МЕТОДИ ЗБЕРІГАННЯ ДАНИХ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ НА ОСНОВІ ЗВ'ЯЗНИХ СПИСКІВ

Анотація. Метою даної роботи є дослідження та порівняльний аналіз методів і структур даних для зберігання інформації рекомендаційної системи, щоб порівняти ефективність їх використання за затратами часу та пам'яті. Вибір методу представлення даних, якими оперує рекомендаційна система, має важливе значення, оскільки ефективний спосіб побудови бази даних для роботи такої системи може зменшити кількість потрібних ресурсів та збільшити кількість доступних алгоритмів для формування списків рекомендацій, а також є важливим з точки зору якості її роботи, швидкості, можливостей масштабування та зручності виконання основних операцій з даними для формування рекомендацій. Наявність 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 в них. У роботі було проведено дослідження різних структур даних, які можна використати для зберігання інформації рекомендаційної системи. Зокрема, таких як зв'язний список, розгорнутий зв'язний список, хеш-таблиця, B-дерево, B+-дерево та бінарна діаграма рішень. Для проведення експериментів з порівняння ефективності застосування різних структур даних за затратами часу та пам'яті було розроблено програмну модель спрощеної рекомендаційної системи, в якій було виділено три основні сутності – агент, сесія та предмет. Найкращі результати показали методи зберігання даних з використанням розгорнутого та інвертованого розгорнутого з'язних списків. Тому було вирішено також провести додаткову серію експериментів з цими структурами даних для різного розміру блоку списку. Розгорнутий список показав кращі результати за використовуваною пам'яттю та за часом генерації сесій. Інвертований розгорнутий список показав перевагу за часом генерації рекомендацій.

Ключові слова: рекомендаційні системи, бази даних, програмна імітаційна модель, зв'язний список, розгорнутий зв'язний список, хеш-таблиця, B-дерево, B+-дерево, бінарні діаграми рішень.

Вступ

Рекомендаційні системи у наш час є важливою складовою соціальних мереж та контент-орієнтованих веб-сайтів і значним чином впливають на те, як користувачі сприймають інформаційний простір у мережі Інтернет [1, 2]. Вибір методу представлення даних, якими оперує рекомендаційна система, має важливе значення, оскільки ефективний 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 бази даних для зберігання інформації рекомендаційної системи є важливою науково-практичною задачею.

На сьогоднішній день 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 по-різному, 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 великої кількості різних методів реалізації баз даних та способів представлення інформації, що можна використати при побудові рекомендаційних систем, викликає необхідність порівняльного аналізу та вибору оптимального методу і структури даних для зберігання інформації у таких системах.

Основна частина

Було проведено дослідження різних структур даних, які можна використати для створення бази даних рекомендаційної системи [15, 16], а саме нижченаведених.

Зв'язний список (linked list) – структура даних, у якій кожен елемент містить поля даних та 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, тому менеджер пам'яті спричиняє значні затримки та накладні витрати пам'яті в процесі роботи.

Розгорнутий зв'язний список (unrolled list) – зв'язний список, кожен елемент якого містить масив логічних елементів. Це дозволяє об'єднати переваги масивів та зв'язних списків. Об'єднання блоків логі-

чних 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 списку гарантується, що незаповненим лишиться не більше одного блоку елементів.

Хеш-таблиця (hash map) – структура даних, у якій пошук елементу здійснюється на основі його ключа. На розташування елемента у хеш-таблиці вказує хеш-значення його ключа. Якщо декілька 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 є найпростішим способом розв'язання колізій, але вимагає використання додаткової пам'яті 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 елементу повторюється, доки не буде досягнуто порожнього запису у таблиці. 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) – структура даних представлена збалансованим та сильно розгалудженим деревом пошуку. Кожен вузол B-дерева, крім листків, є упорядкованим списком, у якому чергуються ключі і вказівники на потомків. Ключі вузла вказують інтервал, у якому знаходяться ключі потомку. **B+-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) – 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.

Для проведення експерименту по вивченню потреб в оперативній пам'яті було розроблено програмну модель спрощеної рекомендаційної системи, в якій було виділено три основні сутності – агент, сесія та предмет [15, 16].

Рекомендаційна система отримує наступні параметри: n_a – кількість агентів, n_s – кількість сесій, n_i – кількість предметів, n_{al} – максимальна кількість вподобань агента, n_{sl} – максимальний розмір сесії.

РС у розробленій програмній моделі працює за наступним алгоритмом:

1. Для кожного з n_a агентів випадковим чином генерується від 1 до n_{al} вподобань. При цьому унікальність вподобань не перевіряється, тому реальна кількість вподобань може виявитися менше.
2. Створюється n_s сесій. До кожної сесії закріплюється випадковим чином обраний агент. Потім серед вподобань цього агента випадковим чином обирається від 1 до $min(n_{al}, n_{sl})$ вподобань, які копіюються до сесії.
3. Випадковим чином обирається контрольна сесія, для якої буде сформовано рекомендацію.
4. Визначаються усі предмети, які належать до контрольної сесії.
5. Здійснюється пошук усіх сесій, вподобання яких мають перетин із вподобаннями контрольної сесії. На цьому етапі є можливість відфільтрувати сесії за розміром перетину.
6. 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.

Результати експериментів наведені у табл. 1, де:

– agentCount – кількість агентів;
– itemsCount – кількість предметів;
– sessionsCount – кількість сесій;
– sessionsSize – розмір сесії;
– maxLikes – максимальна кількість лайків,
– БДР – бінарні діаграми рішень.

Як видно з таблиці 1, найкращі результати показали методи зберігання з використанням розгорнутого та інвертованого розгорнутого з'язних списків. Тому було вирішено також провести додаткову серію експериментів з цими структурами даних для різного розміру блоку (кількість елементів в блоці зв'язного списку), результати якої наведені в табл. 2.

Таблиця 1 –**Результати експериментів для порівняння ефективності різних структур даних для зберігання інформації рекомендаційної системи**

Структура даних	Використана пам'ять, мін. знач., Gb	Використана пам'ять, макс. знач., Gb	Час генерації лайків, ms	Час генерації сесій, ms	Час генерації рекомендацій, ms
agentCount = 65536, itemsCount = 131072, sessionsCount = 262144, sessionSize = 192, maxLikes = 1536					
БДР	0,786	1,3	210945	237214	45766
Хеш-таблиця	2,2	2,2	2312	4034	316
B+-дерево	2,1	2,1	5407	5324	213
Зв'язний список	4,5	4,5	2989	4323	23223
Розгорнутий зв'язний список	0,578	0,578	962	2259	141
Інвертований зв'язний список	0,767	0,767	947	3842	93
agentCount = 131072, itemsCount = 262144, sessionsCount = 524288, sessionSize = 256, maxLikes = 2048					
БДР	1,8	3,6	607610	682754	153045
Хеш-таблиця	6,4	6,4	5359	10811	2312
B+-дерево	5,6	5,6	14925	14554	720
Зв'язний список	12,1	12,1	8052	11195	74346
Розгорнутий зв'язний список	1,2	1,2	2623	5921	412
Інвертований зв'язний список	1,9	1,9	2616	10457	277
agentCount = 262144, itemsCount = 524288, sessionsCount = 1048576, sessionSize = 256, maxLikes = 2048					
БДР	3,4	7,2	1366467	1551391	313122
Хеш-таблиця	12,8	12,8	10661	21722	4248
B+-дерево	11,2	11,2	29810	29111	1357
Зв'язний список	24	24	16051	22323	146845
Розгорнутий зв'язний список	2,4	2,4	5227	11846	808
Інвертований зв'язний список	3,9	3,9	5248	20919	563

Таблиця 2 –**Результати експериментів для порівняння ефективності звичайного та інвертованого розгорнутих зв'язних списків для зберігання інформації рекомендаційної системи**

№ експ.	Структура даних	Кількість елементів в блоці зв'язного списку	Використана пам'ять, макс. знач., Gb	Час генерації лайків, ms	Час генерації сесій, ms	Час генерації рекомендацій, ms
1.	**РЗС**	52	4,7	10954	25355	1998
	ІРЗС		7,1	10845	37742	1563
2.	**РЗС**	130	4,4	10577	24480	1754
	ІРЗС		6,9	10393	42919	1281
3.	**РЗС**	200	5,5	10433	24427	1821
	ІРЗС		8,2	10429	42846	1334
4.	**РЗС**	260	4,8	10404	23710	1602
	ІРЗС		7,8	10933	41616	855

У табл. 2 було використано наступні скорочення: **РЗС** – розгорнутий зв'язний список, **ІРЗС** – інвертований розгорнутий зв'язний список.

Розгорнутий список показав кращі результати за використовуваною пам'яттю в середньому в 1,54 рази та за часом генерації сесій в середньому в 1,68 разів.

В той же час інвертований розгорнутий список показав перевагу за часом генерації рекомендацій в середньому в 1,43 разів.

Час генерації лайків обидва методи показали приблизно однаковим.

Висновки

У роботі було проведено дослідження різних структур даних, які можна використати для зберігання даних рекомендаційної системи. Зокрема, таких як зв'язний список, розгорнутий зв'язний список, хеш-таблиця, B-дерево, B+-дерево та бінарні діаграми рішень.

Для проведення експериментів з порівняння ефективності застосування різних структур даних за затратами часу та пам'яті було розроблено програмну

модель спрощеної рекомендаційної системи, в якій було виділено три основні сутності – агент, сесія та предмет. Найкращі результати показали методи зберігання даних з використанням розгорнутого та інвертованого розгорнутого зв'язних списків. Тому було вирішено також провести додаткову серію експериментів з цими структурами даних для різного розміру блоку списку. Розгорнутий список показав кращі результати за використовуваною пам'яттю в середньому в 1,54 рази та за часом генерації сесій в середньому в 1,68 разів. Інвертований розгорнутий список показав перевагу за часом генерації рекомендацій в середньому в 1,43 разів. Час генерації лайків обидва методи показали приблизно однаковим.

Подальші дослідження будуть спрямовані на дослідження існуючих систем управління базами даних, методів зберігання інформації у них та їх ефективності для рекомендаційних систем.

СПИСОК ЛІТЕРАТУРИ

1. Recommender Systems Handbook (2010) Editors F. Ricci, L. Rokach, B. Shapira, P. B. Kantor, New York, NY, Springer-Verlag New York, Inc., USA. 842 p.
2. Valois B.Jr.C., Oliveira M.A. (2011) Recommender systems in social networks. JISTEM J.Inf.Syst. Technol. Manag., Vol.8 No.3. P. 681-716. URL: https://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752011000300009
3. Фаулер М., Садаладж П. Дж. (2013) NoSQL: Новая методология разработки нереляционных баз данных. Издательский дом «Вильямс», Москва. 192 с.
4. Meier A., Kaufmann M. (2019) SQL & NoSQL Databases. Springer Vieweg, Wiesbaden. P. 201-218. – URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.7089&rep=rep1&type=pdf
5. Cure O., Blin G. (2014) RDF Database Systems: Triples Storage and SPARQL Query Processing. Elsevier Science. 256 p.
6. Yi N., Li C., Feng X., Shi M. (2017) Design and implementation of movie recommender system based on graph database. 14th Web Information Systems and Applications Conference (WISA), IEEE. P. 132-135.
7. Angles R. (2012) A comparison of current graph database models. IEEE 28th International Conference on Data Engineering Workshops, IEEE. P. 171-177.
8. Засядко Г.Е., Карпов А.В. (2017) Проблемы разработки графовых баз данных. Инженерный вестник Дона. №1 (44). URL: https://cyberleninka.ru/article/n/problemy-razrabotki-grafovyh-baz-dannyh
9. Мелков С., Мусатов Д., Савватеев А. (2013) Моделирование социальных сетей. URL: https://kpfu.ru/docs/F117464271/MMS_socnet_cities.pdf
10. Берновски М.М., Кузюрин Н.Н. (2012) Случайные графы, модели и генераторы безмасштабных графов. Труды ИСП РАН. URL: https://cyberleninka.ru/article/n/sluchaynye-grafy-modeli-i-generatory-bezmasshtabnyh-grafov
11. Райгородский А.М. (2012) Математические модели Интернета. "Квант" №4. С. 12-16. – URL: https://elementy.ru/nauchno-populyarnaya_biblioteka/431792
12. Meleshko Ye. (2019) Computer model of virtual social network with recommendation system. Scientific journal Innovative Technologies and Scientific Solutions for Industries, Kharkiv: NURE, Issue 2(8). P. 80-84
13. Робинсон Я., Вебер Д., Эифрем Э. (2016) Графовые базы данных: новые возможности для работы со связанными данными. ДМК Пресс, Москва. 256 с.
14. Neo4j Documentation (2021), Official website of the graph database Neo4j. URL: https://neo4j.com/docs/
15. Міхав В.В., Мелешко Є.В., Якименко М.С. (2020) Метод зберігання даних рекомендаційної системи на основі бінарних діаграм рішень. Системи управління, навігації та зв'язку. ПНТУ, Полтава. Т. 2 (60). С. 85-89.
16. Міхав В.В., Мелешко Є.В., Шимко С.В. (2021) Методи та структури даних для реалізації бази даних рекомендаційної системи соціальної мережі. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація: збірник наукових праць ЦНТУ, Кропивницький. Вип. 4(35). С. 8-16.

Received (Надійшла) 15.10.2021
Accepted for publication (Прийнята до друку) 24.11.2021

The methods of data storing of a recommendation system based on linked lists

V. Mikhav, Ye. Meleshko, M. Yakymenko, D. Bashchenko

Abstract. The goal of this work is to research and comparative analysis of methods and data structures for storing information of a recommendation system in order to compare the effectiveness of their use in terms of time and memory costs. The choice of the method for presenting the data used by the recommendation system is important since an effective way of building a database for the operation of such a system can reduce the amount of resources required and increase the number of available algorithms for generating lists of recommendations, and is also important from the point of view of the quality of its work, speed, scalability, and ease of performing basic operations with data to generate recommendations. The presence of a large number of different methods for implementing databases and presenting information that can be used to build recommendation systems necessitates a comparative analysis and selection of the optimal method and data structure for storing information in them. In the work, research was carried out of various data structures that can be used to store information of the recommender system. In particular, such as linked list, unrolled linked list, hash table, B-tree, B+-tree and binary decision diagram. To carry out experiments comparing the effectiveness of using various data structures in terms of time and memory costs, a computer model of a simplified recommendation system was developed, in which three main entities were distinguished - an agent, a session, and an object. The best results were obtained with data storing methods using unrolled and inverted unrolled linked lists. Therefore, it was decided to also conduct an additional series of experiments with these data structures for different sizes of the list block. The unrolled list showed the best results in terms of memory used and session generation time. The inverted unrolled list showed an advantage in the generation of recommendations.

Keywords: recommendation system, databases, computer simulation model, linked list, unrolled linked list, hash table, B-tree, B+-tree, binary decision diagram.