УДК 004.65+681.518 DOI: https://doi.org/10.32515/2664-262X.2022.5(36).2.130-136

В.В. Міхав, асп., **Є.В. Мелешко,** проф., д-р техн. наук, М**.С. Якименко,** доц., канд. физ.-мат. наук, **Я.П. Шуліка,** асп.
Центральноукраїнський національний технічний університет, м. Кропивницький, Україна
e-mail: mihaw.wolodymyr@gmail.com, elismeleshko@gmail.com, m.yakymenko@gmail.com, yar.shulika@gmail.com

Розробка системи управління базою даних рекомендаційної системи для комп’ютерних та комп’ютерно-інтегрованих систем

Метою даної роботи є розробка системи управління базою даних рекомендаційної системи для комп’ютерних та комп’ютерно-інтегрованих систем, та порівняння якості її роботи з існуючими системами. На сьогоднішній день рекомендаційні системи мають широке застосування у комп’ютерних мережах, зокрема, в соціальних мережах, системах Інтернет-торгівлі, поширенні медіа-контенту, реклами тощо, а також у комп’ютерно-інтегрованих системах, зокрема, в Інтернеті речей та розумних будинках. Ефективний спосіб представлення даних, необхідних для роботи рекомендаційної системи, може зменшити кількість потрібних ресурсів та полегшити розробку і використання більш складних алгоритмів для формування списків рекомендацій. У цій роботі було проведено дослідження можливості та ефективності використання відкритих лінійних списків для збереження даних рекомендаційної системи. Розроблена система значно випереджає розглянуті існуючі інструменти як за швидкістю роботи, так і за ефективністю використання пам’яті.
бази даних, системи управління базами даних, рекомендаційні системи, комп’ютерні системи, комп’ютерно-інтегровані системи, лінійні списки

Постановка проблеми. На сьогоднішній день рекомендаційні системи мають широке застосування у комп’ютерних мережах [1-5], зокрема, в соціальних мережах, системах Інтернет-торгівлі, поширенні медіа-контенту, реклами тощо, а також у комп’ютерно-інтегрованих системах [6-8], зокрема, в Інтернеті речей IoT та розумних будинках. Ефективний спосіб представлення даних, необхідних для роботи рекомендаційної системи, може зменшити кількість потрібних ресурсів та полегшити розробку і використання більш складних алгоритмів для формування списків рекомендацій. У цій роботі було проведено дослідження можливості та ефективності використання відкритих лінійних списків для збереження даних рекомендаційної системи.

Аналіз останніх досліджень і публікацій. Бази даних надають можливість зберігати і обробляти інформацію у зручному для програміста та користувачів форматі. Існують різні моделі збереження інформації у системах управління базами даних [9-13]:

1. *Прості моделі даних.* Дані зберігаються у звичайних файлах. Наприклад, у електронних таблицях – у файлах формату .csv або інших. Підходять для зберігання даних з простою структурою.
2. *Ієрархічні моделі даних.* Можуть бути представлені файловою системою, за дорогою якої можна здійснити деревовидне структурування даних шляхом розміщення їх у каталогах та файлах. Кожен запис може мати не більше одного батьківського запису і не можливо реалізувати відношення «багатьох-до-багатьох». Прикладом таких баз даних можуть бути різні файлові системи.

3. *Реляційні моделі даних*. Найпопулярніші моделі загального призначення. Дані представлені у вигляді таблиць. Окремий запис представлений у вигляді рядка таблиці. Для доступу до даних використовується структурована мова запитів SQL. Ці моделі мають ряд недоліків, що призводять до надмірності даних, які можуть приводити до різних аномалій та порушень цілісності даних. Для усунення цих недоліків структури використовується нормалізація. Приклади – MySQL, SQLite, Visual FoxPro, Access, Oracle.

4. *Модель даних «ключ-значення»*. Належать до NoSQL типу. Для зберігання даних використовуються хеш-таблиці, які зберігають колекцію записів, що містять множину різних полів даних. Записи зберігаються та вилучаються з використанням ключа. Немає жорсткої схеми відношень між даними. Хеш-таблиці забезпечують швидкий доступ до даних, але ресурсомісткі. Найпопулярнішим прикладом є БД Redis.

5. *Документні моделі даних*. Належать до NoSQL типу. В основі лежать документні сховища, що мають структуру дерева. Дані зберігаються в структурованих форматах JSON, BSON або XML. Кожен документ може мати свою внутрішню структуру. Запити до бази даних дозволяють знайти документ або частину документу. Документні моделі даних застосовуються в системах управління змістом, видавничій справі, документальному пошуку. Приклади – CouchDB, MongoDB, RethinkDB, eXist.

6. *Графові моделі даних*. Належать до NoSQL типу. Дані зберігаються у вигляді вершин та ребер графів. Вершини та ребра можуть мати будь-яку кількість пов’язаних з ними властивостей. Часто застосовують для моделювання соціальних графів, семантичної павутини тощо. Можуть бути високопродуктивними, є більш наглядними та простими для внесення змін. Приклади – Neo4j, JanusGraph, JanusGraph, ArangoDB.

7. *Комбіновані моделі даних*. Об’єднують переваги SQL та NoSQL підходів у побудові баз даних. Мають можливість горизонтального масштабування, велику продуктивність, гнучкість, але велику ресурсомісткість і необхідність спеціалізованих знань для роботи з ними. Приклади – MemSQL, VoltDB, Calvin, Calvin.

Вибір моделі збереження даних залежить від властивостей даних, що потребують обробки (зокрема, складності їх структури), сфери застосування і апаратних можливостей.

Все частіше для зберігання даних рекомендаційних систем та інших додатків починають використовувати графові моделі [14, 15]. І це відбувається через ряд переваг графових моделей, зокрема, зручності представлення даних для програміста та можливості легкого масштабування і створення мережевих баз даних. Яскравим прикладом такого підходу являється побудова рекомендаційних систем з застосуванням графової СУБД Neo4j [13]. 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.

При зберіганні даних рекомендаційної системи одними з важливих параметрів бази даних є швидкість операцій читання/запису інформації, а також необхідний об’єм пам’яті для збереження даних у тому чи іншому форматі. Тому для рекомендаційних систем часто доцільно використовувати прості моделі даних. У роботі [16] було досліджено можливості та ефективність застосування різних структур даних для збереження інформації рекомендаційної системи. Зокрема, таких як зв’язний список, розгорнутий зв’язний список, хеш-таблиця, B-дерево, B+-дерево та бінарні діаграми рішень. Найкращі результати за використаною пам’яттю та швидкістю виконання

операцій показали зв'язні лінійні списки.

Постановка завдання. Таким чином, метою даної роботи є розробка системи управління базою даних рекомендаційної системи з використанням лінійних списків для комп'ютерних та комп'ютерно-інтегрованих систем та порівняння якості її роботи з існуючими системами.

Виклад основного матеріалу. У цій роботі було розроблено систему управління базою даних для рекомендаційної системи та проведено дослідження ефективності використання відкритих лінійних списків для збереження інформації рекомендаційної системи.

Лінійні списки бувають різних видів і класифікуються за способами реалізації.

Зв'язний список – 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 – 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.

V-список – це розгорнутий зв'язний список, у якому розмір блоків задається геометричною прогресією. Це дозволяє зменшити зважений час додавання елементу і зменшити накладні витрати у списку малої розмірності, але збільшує максимальні накладні витрати у списку великої розмірності.

Ми пропонуємо зберігати у розгорнутому списку рекомендації користувачам для кожної сесії роботи рекомендаційної системи. У варіанті із інвертованим списком ми також зберігаємо дані про вподобання користувачів для кожного об'єкту.

Завдяки інвертованому списку ми можемо використати наступну схему вибірки даних: 1) для кожного вподобання користувача у сесії вибрати з інвертованого списку усі сесії, пов'язані з об'єктом вподобання; 2) вибрати вподобання користувачів усіх відібраних сесій.

Таким чином це дозволяє переглядати лише частину даних. Завдяки цьому вибірка для формування рекомендації виконується швидше, але додавання нових вподобань вимагає більше обчислень та пам'яті.

У проведених експериментах, розгорнутий список показав найкращі показники швидкодії та використання пам'яті. Профілювання показало, що 75% часу роботи тесту для варіанту з розгорнутим списком зайняло генерування випадкових даних для тесту, 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.

Для перевірки ефективності запропонованого методу представлення даних було проведено експерименти з таким програмним забезпеченням:

- реляційна система керування базами даних Postgresql;
- резидентне сховище пар «ключ-значення» Redis;

– графова база даних Neo4j.

Кожен метод представлення даних було перевірено за наступними показниками:

– час заповнення сховища тестовими даними;
– об’єм пам’яті, зайнятий сховищем після заповнення;
– час генерації рекомендації.

В якості тестових даних використано набір даних MovieLens. Об’єм даних у форматі .csv складає 724 Мб. Результати проведених експериментів наведені у табл.1.

Час завантаження даних до програми, розробленої на основі нашого методу, склав 1.3 секунди, при цьому програма використала 616 Мб пам’яті. Генерація рекомендації для користувача зайняла 0.15 секунди.

У PostgreSQL було створено таблицю для представлення вподобань, яка складалася з полів для ідентифікаторів користувача та об’єкту. Було перевірено роботу без використання індексів та з використанням індексів для кожного поля. Додавання даних відбувалося за допомогою SQL-інструкцій, при цьому було відзначено, що час додавання зменшувався при збільшенні кількості даних на один запит. Запис даних блоками по 100000 рядків склав 44 секунди у таблицю без індексів та 113 секунд у таблицю з індексами, тоді як запис блоками по 10000 – 58 та 125 секунд відповідно. Об’єм пам’яті, зайнятої таблицею, склав 786 Мб, індекси зайняли 331 Мб. Для побудови рекомендації було використано два варіанти SQL-запиту. У першому варіанті запиту було поєднано вирази IN та JOIN, час його виконання склав 13.9 секунд для таблиці без індексів та 16.3 секунд для таблиці з індексами. У другому варіанті запиту було використано лише JOIN, час виконання склав 14.5 секунд для таблиці без індексів та 6.7 секунд для таблиці з індексами.

У Redis для збереження даних було застосовано вбудований тип Set (множина). Для кожного користувача було створено множину з його вподобаннями, а для кожного об’єкту – множину з користувачами, які його вподобали. Redis здійснює запис даних асинхронно, і створення великої кількості одночасних запитів може вичерпати доступні ресурси. Тому великі об’єми даних потрібно завантажувати блоками, додаючи затримку після кожного блоку. Заповнення зайняло 89 секунд, для представлення даних Redis використав 1.4 Гб. Вибірка даних для генерації рекомендації зайняла 27 секунд.

Для завантаження даних до Neo4j було використано вбудовану функцію імпорту з .csv файлу. Завантаження вподобань зайняло 335 секунд, база даних зайняла 1.7 Гб. Для підвищення швидкодії було створено індекси по зв’язках між сутностями. Генерація рекомендацій зайняла 155 секунд.

Таблиця 1 – Порівняння якості роботи запропонованої та існуючих систем управління базами даних

Інструмент	**Час заповнення, с**	**Об’єм пам’яті, Гб**	**Час генерації рекомендації, с**
PostgreSQL	113	1.09	6.7
Redis	89	1.4	27
Neo4j	335	1.7	155
Розроблена система на основі лінійних списків	1.3	0.6	0.15

Джерело: розроблено авторами

Висновки. У даній роботі розроблено систему управління базою даних рекомендаційної системи на основі лінійних списків, яку можна використати для

комп'ютерних мереж та комп'ютерно-інтегрованих систем.

Для перевірки ефективності запропонованого методу представлення даних у рекомендаційній системі було проведено порівняльні експерименти з таким програмним забезпеченням як: реляційна СУБД Postgresql, резидентна СУБД Redis та графова СУБД Neo4j. Кожен метод представлення даних було перевірено за наступними показниками: час заповнення сховища тестовими даними; об'єм пам'яті, зайнятий сховищем після заповнення; час генерації рекомендації. Розроблена система значно випереджає розглянуті існуючі інструменти як за швидкістю роботи, так і за ефективністю використання пам'яті. Варто зауважити, що вона не передбачає витіснення даних на диск чи мережевий доступ до сховища, тому необхідно мати додаткове постійне сховище для збереження даних про вподобання.

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. Anitha J., Kalaiarasu M. (2022), "Retraction Note to: Optimized machine learning based collaborative filtering (OMLCF) recommendation system in e-commerce", J Ambient Intell Human Comput, DOI: https://doi.org/10.1007/s12652-022-04093-4
3. Priya A.S.B. (2022), "Bhuvaneswaran, R.S. Retraction Note to: Cloud service recommendation system based on clustering trust measures in multi-cloud environment", J Ambient Intell Human Comput, DOI: https://doi.org/10.1007/s12652-022-04056-9
4. Paul, D., Kundu, S. (2020), "A Survey of Music Recommendation Systems with a Proposed Music Recommendation System", In: Mandal, J., Bhattacharya, D. (eds) Emerging Technology in Modelling and Graphics, Advances in Intelligent Systems and Computing, Vol. 937, Springer, Singapore, DOI: https://doi.org/10.1007/978-981-13-7403-6_26
5. 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
6. Felfernig A., Polat-Erdeniz S., Uran C. et al. (2019), "An overview of recommender systems in the internet of things", J Intell Inf Syst 52, P. 285-309, DOI: https://doi.org/10.1007/s10844-018-0530-7
7. Nawara D., Kashef R. (2020), "IoT-based Recommendation Systems – An Overview", 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp. 1-7, DOI: https://doi.org/10.1109/IEMTRONICS51293.2020.9216391
8. Bouazza H., Said B., Laallam F.Z. (2022), "A hybrid IoT services recommender system using social IoT", Journal of King Saud University – Computer and Information Sciences, DOI: https://doi.org/10.1016/j.jksuci.2022.02.003, URL: https://www.sciencedirect.com/science/article/pii/S1319157822000362
9. Hills T. (2016), "NoSQL and SQL Data Modeling. Bringing Together Data, Semantics, and Software", Technics Publications, 260 p.
10. 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
11. Cure O., Blin G. (2014), "RDF Database Systems: Triples Storage and SPARQL Query Processing", Elsevier Science, 256 p.
12. Robinson I., Webber J., Eifrem E. (2016), "Graph Databases: New Opportunities for Connected Data", O'Reilly Media, 238 p.
13. "Neo4j Documentation" (2021), Official website of the graph database Neo4j, URL: https://neo4j.com/docs/
14. 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, IEEE, P. 132-135.
15. Angles R. (2012), "A comparison of current graph database models", IEEE 28th International Conference on Data Engineering Workshops, IEEE, P. 171-177.
16. Міхав В.В., Мелешко Є.В., Якименко М.С., Бащенко Д.В. (2021), "Методи зберігання даних рекомендаційної системи на основі зв'язних списків", Системи управління, навігації та зв'язку, Т. 4(66), Полтава, С. 59-62. – DOI: https://doi.org/10.26906/SUNZ.2021.4.059

Referencis

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. Anitha, J. & Kalaiarasu, M. (2022), "Retraction Note to: Optimized machine learning based collaborative filtering (OMLCF) recommendation system in e-commerce", J Ambient Intell Human Comput, DOI: https://doi.org/10.1007/s12652-022-04093-4
3. Priya A.S.B. (2022), "Bhuvaneswaran, R.S. Retraction Note to: Cloud service recommendation system based on clustering trust measures in multi-cloud environment", J Ambient Intell Human Comput, DOI: https://doi.org/10.1007/s12652-022-04056-9
4. Paul, D. & Kundu, S. (2020). "A Survey of Music Recommendation Systems with a Proposed Music Recommendation System", In: Mandal, J., Bhattacharya, D. (eds) Emerging Technology in Modelling and Graphics, Advances in Intelligent Systems and Computing, Vol. 937, Springer, Singapore, DOI: https://doi.org/10.1007/978-981-13-7403-6_26
5. 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, Retrieved from https://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752011000300009
6. Felfernig, A., Polat-Erdeniz, S., Uran, C. et al. (2019). "An overview of recommender systems in the internet of things", J Intell Inf Syst 52, P. 285-309, DOI: https://doi.org/10.1007/s10844-018-0530-7
7. Nawara, D. & Kashef, R. (2020). "IoT-based Recommendation Systems – An Overview", 2020 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS), pp. 1-7, DOI: https://doi.org/10.1109/IEMTRONICS51293.2020.9216391
8. Bouazza, H., Said, B. & Laallam, F.Z. (2022). "A hybrid IoT services recommender system using social IoT", Journal of King Saud University – Computer and Information Sciences, DOI: https://doi.org/10.1016/j.jksuci.2022.02.003, Retrieved from https://www.sciencedirect.com/science/article/pii/S1319157822000362
9. Hills T. (2016). "NoSQL and SQL Data Modeling. Bringing Together Data, Semantics, and Software", Technics Publications, 260 p.
10. 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
11. Cure, O. & Blin, G. (2014). "RDF Database Systems: Triples Storage and SPARQL Query Processing", Elsevier Science, 256 p.
12. Robinson I., Webber J., Eifrem E. (2016), "Graph Databases: New Opportunities for Connected Data", O'Reilly Media, 238 p.
13. "Neo4j Documentation" (2021), Official website of the graph database Neo4j, Retrieved from https://neo4j.com/docs/
14. 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, IEEE, P. 132-135.
15. Angles R,. (2012), "A comparison of current graph database models", IEEE 28th International Conference on Data Engineering Workshops, IEEE, P. 171-177.
16. Mikhav, V.V., Meleshko, Ye.V., Yakymenko, M.S. & Bashchenko, D.V. (2021). "The methods of data storingof a recommendation system based on linked lists", Control, navigation and communication systems, *Vol. 4(66),* 59-62. DOI: https://doi.org/10.26906/SUNZ.2021.4.059 [in Ukrainian]

Volodymyr Mikhav, post-graduate**, Yelyzaveta Meleshko,** Prof., DSc., **Mykola Yakymenko,** Assos. Prof., PhD phys.&math. sci., **Yaroslav Shulika,** post-graduate
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine

Development of a Database Management System of Recommendation Systems for Computer Networks and Computer-integrated Systems

The goal of this work is to develop a database management system of the recommendation system for computer networks and computer-integrated systems, as well as to compare the quality of its work with existing systems.

Today, recommendation systems are widely used in computer networks, in particular, in social networks, Internet commerce systems, media content distribution, advertising, etc., as well as in computer-integrated systems, in particular, in the Internet of Things and smart houses. An effective way to present the data required for the recommendation system can reduce the number of resources required and facilitate the development and use of more sophisticated algorithms for compiling lists of recommendations. When storing data from the recommendation system, one of the important parameters of the database is the speed of reading/writing

information, as well as the amount of memory required to store data in one format or another. Therefore, it is advisable to use simple data models. This paper investigated the feasibility and effectiveness of using open linear lists to store recommendation system data in computer networks and computer-integrated systems. To test the effectiveness of the proposed method of presenting data in the recommendation system, comparative experiments were conducted with such software as: relational database management system Postgresql, resident repository key-value pairs Redis and graph database Neo4j. Each method of presenting data was tested on the following indicators: time of filling the repository with test data; the amount of memory occupied by the repository after filling; recommendation generation time. The MovieLens data set was used as test data.

The developed database management system based on linear lists is significantly ahead of the existing tools in terms of both speed and efficiency of memory use.

databases, database management systems, recommendation systems, computer systems, computer-integrated systems, linear lists

Одержано (Received) 22.04.2022 *Прорецензовано (Reviewed) 30.04.2022*
Прийнято до друку (Approved) 30.05.2022