КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

УДК 004.9 DOI: https://doi.org/10.32515/2664-262X.2021.4(35).8-16

В.В. Міхав, асп., **Є.В. Мелешко,** доц., д-р техн. наук, **С.В. Шимко,** асп.
Центральноукраїнський національний технічний університет, м.Кропивницький, Україна
e-mail: mihaw.wolodymyr@gmail.com, elismeleshko@gmail.com, shymko97@gmail.com

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

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.

рекомендаційні системи, бази даних, структури даних, програмна імітаційна модель

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 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 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 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) – 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.

Таблиця 1 – Результати серії експериментів для порівняння часу формування рекомендацій системою при використанні різних структур даних для реалізації бази даних

№ серії експ.	**BDD, мс**	**Hash map, мс**	**Linked list, мс**	**B+ tree, мс**	**Unrolled list, мс**	**Unrolled list + inverted, мс**
1	45766	316	23223	213	141	93
2	153045	2312	74346	720	412	277
3	313122	4248	146845	1357	808	563
4	643864	9153	-	2797	1602	855

Примітка. Прочерк у комірці таблиці показує, що не можливо було отримати результат через нестачу оперативної пам’яті.

Джерело: розроблено автором

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.

Рисунок 1 – Час генерації рекомендацій системою для кожної з розглянутих структур даних
Джерело: розроблено автором

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.

Таблиця 2 – Результати серії експериментів для порівняння кількості використаної пам'яті для формування рекомендацій системою при використанні різних структур даних для реалізації бази даних

№ серії експ.	BDD, Гб	Hash map, Гб	Linked list, Гб	B+ tree, Гб	Unrolled list, Гб	Unrolled list + inverted, Гб
1	1,3	2,2	4,5	2,1	0,578	0,767
2	3,6	6,4	12,1	5,6	1,2	1,9
3	7,2	12,8	24	11,2	2,4	3,9
4	14,4	25,5	-	22,3	4,8	7,8

Примітка. Прочерк у комірці таблиці показує, що не можливо було отримати результат через нестачу оперативної пам'яті.
Джерело: розроблено автором

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.

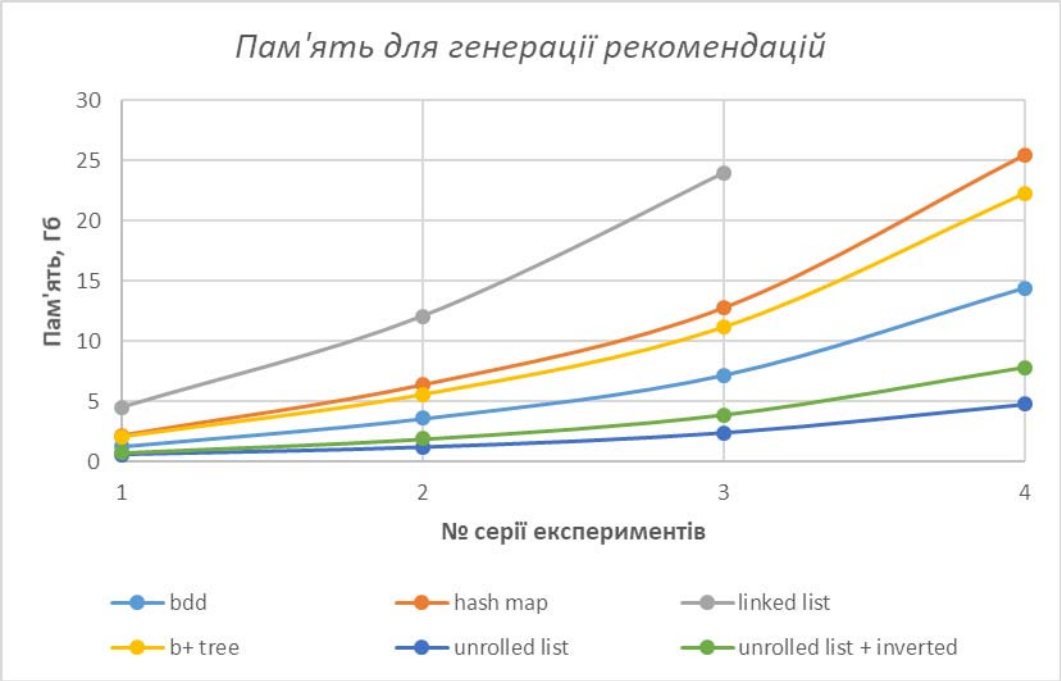

Рисунок 2 – Використана пам’ять для генерації рекомендацій системою для кожної з розглянутих структур даних

Джерело: розроблено автором

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.

Список літератури

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. №1(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. №4. 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. 80-84.
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. №3.2. pp. 156-170.
17. Кнут Д.Э. Искусство программирования, Том 4А. Комбинаторные алгоритмы. Часть 1. М.: Вильямс. 2013. 960 с.

Referencis

1. Editors Ricci, F., Rokach, L., Shapira, B. & Kantor, P.B. (2010). *Recommender Systems Handbook*. New York, NY, USA: Springer-Verlag New York, Inc [in English].
2. Valois, B.Jr.C. & Oliveira, M.A. (2011). Recommender systems in social networks. *JISTEM J.Inf.Syst. Technol. Manag., Vol.8, No.3.* 681-716. Retrieved from https://www.scielo.br/scielo.php?script=sci_arttext&pid=S1807-17752011000300009 [in English].
3. Fauler, M., Sadaladzh, P. Dzh. (2013). *NoSQL: Novaja metodologija razrabotki nereljacionnyh baz dannyh* [*NoSQL: A new methodology for the development of non-relational databases*]. Williams Publishing House, Moscow. [in Russian].
4. Meier, A. & Kaufmann, M. (2019). SQL & NoSQL Databases. *Springer Vieweg, Wiesbaden.* 201-218. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.468.7089&rep=rep1&type=pdf [in English].
5. Cure, O., Blin, G. (2014). RDF Database Systems: Triples Storage and SPARQL Query Processing. *Elsevier Science* [in English].
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.* 132-135[in English].
7. Angles, R. (2012). A comparison of current graph database models. *IEEE 28th International Conference on Data Engineering Workshops, IEEE,* 171-177 [in English].
8. Zasjadko, G.E. & Karpov, A.V. (2017). Problemy razrabotki grafovyh baz dannyh [Problems in the development of graph databases]. *Inzhenernyj vestnik Dona – Engineering journal of Don. №1 (44).* Retrieved from https://cyberleninka.ru/article/n/problemy-razrabotki-grafovyh-baz-dannyh [in Russian].
9. Melkov, S., Musatov, D. & Savvateev, A. (2013) Modelirovanie social'nyh setej [Social networks' modeling]. Retrieved from https://kpfu.ru/docs/F117464271/MMS_socnet_cities.pdf [in Russian].
10. Bernovski, M.M. & Kuzjurin, N.N. (2012). Sluchajnye grafy, modeli i generatory bezmasshtabnyh grafov [Random Graphs, Models, and Scaleless Graph Generators]. *Trudy ISP RAN – Proceedings of the Institute for System Programming of the Russian Academy of Sciences.* 419-432. Retrieved from https://cyberleninka.ru/article/n/sluchaynye-grafy-modeli-i-generatory-bezmasshtabnyh-grafov [in Russian].
11. Rajgorodskij, A.M. (2012). Matematicheskie modeli Interneta [Mathematical models of the Internet]. *"Kvant" – "Quantum", No 4.* 12-16. Retrieved from https://elementy.ru/nauchno-populyarnaya_biblioteka/431792 [in Russian].
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).* 80-84 [in English].
13. Robinson, Ja., Veber, D., Jeifrem, Je. (2016). *Grafovye bazy dannyh: novye vozmozhnosti dlja raboty so svjazannymi dannymi* [*Graph databases: new possibilities for working with related data*]. Moskow: DMK Press [in Russian].
14. Neo4j Documentation (2020). *neo4j.com.* Retrieved from https://neo4j.com/docs [in English].
15. Aho, A., Hopcroft, J., Ullman, J. (2000). *Struktury dannyh i algoritmy* [*Data Structures and Algorithms*]. Williams Publishing House, Moscow. [in Russian].
16. Minato, S. (2001). Zero-suppressed BDDs and their applications. *International Journal on Software Tools for Technology Transfer, №3.2*. 156-170 [in English].
17. Knuth, D.E. (2013). *Iskusstvo programmirovanija* [*The art of programming*], Tom 4A. Kombinatornye algoritmy. Chast' 1 [Art of Computer Programming, Volume 4A, The: Combinatorial Algorithms, Part 1]. Williams Publishing House, Moscow. [in Russian].

Volodymyr Mikhav, post-graduate**, Yelyzaveta Meleshko,** Assoc. Prof., DSc., **Serhii Shymko,** post-graduate
Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine

Methods and Data Structures for Implementing a Database of Social Networks' Recommendation Systems

The goal of this work is to research and program implementation of methods and data structures for building a database of a recommendation system in order to compare the efficiency of their use in terms of time and memory costs. The presence of a large number of different methods of database implementation necessitates a comparative analysis and selection of the optimal method and data structure for storing information in recommendation systems.

A research on various data structures that can be used to create a recommendation system database, in particular, the linked list, unrolled linked list, hash table, B-tree, B+-tree, and binary decision diagram were examined was conducted. A series of experiments on a software simulation model of a recommendation system with a different number of agents, items and sessions was also carried out.

The following research results were obtained. According to the results of the experiments, the unrolled linked list showed the best time and memory effectiveness. The B+-tree structure showed results close to a hash table. The access time to an individual element is stable in both cases, but the B+-tree has certain advantages – the elements are kept sorted, and when resizing, there is no need to expand the memory area. The worst results were shown by the data structure of the binary decision diagram, both in terms of time consumption and memory consumption. Profiling showed that 75% of the test run time for the option with an unrolled list was taken by generating random data for software simulation of agents and items of the recommendation system, therefore, the data warehouse itself has high performance indicators. Profiling of the variant with an inverted list showed that access to random blocks takes longer due to the inability to cache them, therefore, under real load conditions, the time for inserting new data will be longer, and the relative efficiency of using the inverted list will increase. For the most efficient use of memory, the block size of the linked list should be adapted so that the blocks are as full as possible. Small blocks reduce memory waste, but increase the time to traverse all the elements of the list and increase memory overhead.
recommendation systems, databases, data structures, computer simulation model

Одержано (Received) 10.04.2021 *Прорецензовано (Reviewed) 18.04.2021*
Прийнято до друку (Approved) 26.04.2021

УДК 004.8/681.5 DOI: https://doi.org/10.32515/2664-262X.2021.4(35).16-23

Р.М. Минайленко, доц., канд. техн. наук, **О.Г. Собінов,** викл., **О.К. Конопліцька-Слободенюк,** викл., **К.О. Буравченко**, канд. техн.наук
Центральноукраїнський національний технічний університет, Кропивницький, Україна
e-mail: aron70@ukr.net, sagcob14@gmail.com

Архітектурні особливості систем розподілених обчислень

В статті проведено аналіз архітектурних особливостей систем розподілених обчислень Головним завданням, яке вирішують технології розподілених обчислень є забезпечення доступу до глобально розподілених ресурсів і вирішення задач, що потребують значних обчислювальних потужностей та не можуть бути реалізовані на звичайному комп'ютері. Складність реалізації глобальних задач обумовлена тим, що доступ до необхідних даних може відбуватись на різних комп'ютерах. Крім того, розподілені обчислювальні системи, які формуються із автономних ресурсів, можуть змінювати свою архітектуру динамічно. Керування такими розподіленими обчислювальними системами потребує пошуку нових моделей обчислення і пошуку архітектурних рішень для побудови нових систем які б відповідали сучасному рівню розвитку інформаційних технологій.
комп'ютер, розподілені обчислення, інформаційні технології, архітектурні особливості

Постановка проблеми. Останнім часом спостерігається все більше проникнення інформаційних технологій майже у всі галузі життєдіяльності людства. Розвиток інформаційних технологій пов'язаний з виникненням нових задач, що потребують значних обчислювальних ресурсів і не можуть бути вирішені на звичайному комп'ютері.

Великий об'єм обчислень потребує створення, так званих, суперкомп'ютерів, що реалізувати технічно не завжди можливо. Та існує і інший спосіб вирішення вказаної проблеми, коли складна задача розподіляється на певну кількість підзадач, що виконуються паралельно. І тут стають у пригоді системи розподіленого обчислення.