УДК 004.652

КАРКАСНА СХЕМА БАЗИ ДАНИХ ПОЛІГОНУ КРАЙОВИХ ЗАДАЧ МАТЕМАТИЧНОЇ ФІЗИКИ

Пасенченко Т. О.[1]
[1] Одеський національний університет імені І. І. Мечникова
[1] Україна, Одеса
[1] tomas.pasenchenko@gmail.com

Анотація. *Для полігону крайових задач математичної фізики побудовано базу даних результатів запусків на основі каркасної моделі даних. У СКБД SQLite каркасну схему порівняно зі схемою із сурогатними ключами за однакової фізичної організації відношення запусків. Каркасна схема виконує зведений запит у 1,6–1,7 раза, точковий у 1,1–1,2 раза, а запит запису результатів нового запуску у 1,1–1,35 раза швидше, проте розмір файлу БД у цьому випадку на 20 % більший через ширший ключ відношення запусків.*

Ключові слова: *база даних; чисельні методи; каркасна модель даних; SQLite; експеримент; оптимізація продуктивності; штучний інтелект.*

Вступна частина. Високоточне розв'язання крайових задач динамічної теорії пружності методом сингулярних інтегральних рівнянь [1] потребує великої кількості запусків розрахункових програм на кластері MPI та на графічному прискорювачі CUDA. Сукупність таких задач з еталонними розв'язками утворює полігон для перевірки програм і порівняння паралельних платформ. Кожен запуск визначається опціями, серед яких тип задачі KZ, тип падаючої хвилі NPR, кількість об'єктів MU, кількість точок колокації N і точність арифметики KREAL. Результати запусків мають бути придатними для пошуку та повторного використання [2], тому їх зберігають у реляційній базі даних (БД), зокрема у вбудованій СКБД SQLite, яку дедалі частіше застосовують і для аналітичних запитів [3].

Типова схема такої БД має сурогатний ключ (ID) у кожному відношенні. Сурогатний ключ не несе змісту предметної області, тому запит за значеннями опцій потребує з'єднання (JOIN) кількох відношень, а запис результатів нового запуску – попереднього пошуку сурогатних ключів. Каркасна модель даних (КМД) [4] пропонує інший шаблон схеми, у якому ключем кожного відношення є конкатенація ключів атомарних сутностей-класів, а більшість запитів зводиться до індексної вибірки з одного відношення. Каркасний аналіз предметної області методу сингулярних інтегральних рівнянь виконано в [5], а CASE-оболонку, що керується метаданими каркасної схеми, описано в [6].

Метою роботи є побудова БД конфігурацій і запусків полігону на основі КМД та числове порівняння цієї БД зі схемою із сурогатними ключами за часом виконання запитів і запису та обсягом файлу бази даних.

Основна частина. Каркасна схема містить шість словників, якими є відношення атомарних сутностей-класів з унарним ключем. Це KZ (вільні отвори або нерухомі жорсткі шайби), NPR (чотири типи падаючої хвилі), MU, N, KREAL (REAL*8 або REAL*16) і KPL (платформа MPI або CUDA). Коди перших п'яти словників збігаються з кодами опцій розрахункових програм. Складені сутності-класи подано відношеннями KONFIG (KZ, NPR, MU, N, NAZVA) і PROGIN (KZ, NPR, MU, N, KREAL, KPL, NOM, T), де підкреслено ключові атрибути, NOM є номером повтору запуску, а T є часом розв'язання. Ключ запуску збігається з набором опцій, за яким запуск можна відтворити. Нову опцію вводять як ще один слов-

ник і ще один ключовий атрибут, що відповідає модифікованості каркасної схеми [4]. Відношення каркасної схеми перебувають у п’ятій нормальній формі.

Схема із сурогатними ключами зберігає ті самі дані в шести відношеннях. Словники zadacha, hvylia, tochnist і platforma та відношення konfig мають сурогатний ключ id, а їхні природні ключі оголошено унікальними. Відношення запусків progin має ключ (konfig_id, tochnist_id, platforma_id, nom). Щоб порівнювати вид ключів, а не спосіб зберігання, відношення запусків в обох схемах зберігається як кластеризоване B-дерево за первинним ключем (WITHOUT ROWID). Обидві схеми нормалізовано і мають індекси для всіх умов запитів, тож різняться вони лише видом ключів.

Програма, що написана мовою Python 3.13, породжує словники каркасної БД з метаданих, як це робить інсталятор CASE-оболонки [6], створює обидві БД у SQLite 3.53 і заповнює їх однаковими модельними даними. Для MU від 1 до m, семи значень N від 21 до 301, трьох допустимих поєднань точності та платформи (REAL*16 лише на MPI) і п’яти повторів отримано $z = 840m$ запусків (кортежів відношення PROGIN), де m змінювалося від 4 до 1190. Час T заповнено випадковими числами, оскільки його значення не впливають на час доступу.

Виміряно три запити. Зведений запит повертає середній час розв’язання за кожним N, точністю і платформою для заданих KZ, NPR і MU, що дає 21 рядок, обчислений за 105 кортежами. Точковий запит повертає п’ять повторів одного запуску. Запит запису вставляє результати нового запуску за значеннями опцій, як це робить розрахункова програма після завершення. У каркасній БД обидва запити звертаються лише до відношення PROGIN за префіксом його ключа, а у схемі із сурогатними ключами потребують з’єднання шести відношень. Перевірку зовнішніх ключів під час запису увімкнено в обох БД, але схема із сурогатними ключами додатково потребує пошуку сурогатних ключів у konfig і словниках. Для кожного z виконано 40 повторень по 5000 операцій з випадковими ключами, причому повторення для двох схем чергувалися, щоб зміна навантаження машини впливала на обидві схеми однаково. Наведено медіани часу і медіану відношення часу з міжквартильним інтервалом. Перевірено, що обидві БД повертають однакові відповіді і що плани запитів не містять повного перегляду відношень.

Результати виконання обчислювального експерименту наведено на рис. 1. Зведений запит у каркасній БД виконується за 33–45 мкс проти 56–69 мкс, тобто в 1,6–1,7 раза швидше за всіх z (міжквартильний інтервал 1,5–1,9). Схема із сурогатними ключами потребує п’яти з’єднань і тимчасового B-дерева для групування, тоді як у PROGIN кортежі вже впорядковано за ключем, префікс якого задає умова запиту, а решта атрибутів – групування. Точковий запит виконується за 6,2–11,3 мкс проти 7,1–13,0 мкс, тобто в 1,1–1,2 раза швидше, оскільки каркасна схема не потребує пошуку в словниках і konfig. Запит запису результатів нового запуску в каркасній БД швидший у 1,3 раза при $z \le 3 \cdot 10^4$ і в 1,1 раза при $z \approx 10^6$, бо зі зростанням z частка сталого часу пошуку сурогатних ключів у часі вставлення до B-дерева зменшується. Ціною природного ключа є обсяг: при $z = 999\,600$ файл каркасної БД має 34,1 МБ проти 28,5 МБ, тобто є на 20 % більшим, бо ключ відношення запусків містить сім атрибутів замість чотирьох. Водночас KONFIG займає 3,3 МБ проти 4,4 МБ для konfig разом з індексом природного ключа, оскільки сурогатний ключ потребує другої індексної структури. Виміряний час містить практично однакові для обох схем накладні витрати інтерпретатора Python, тому відношення часу виконання в самій СКБД є не меншим за наведене.

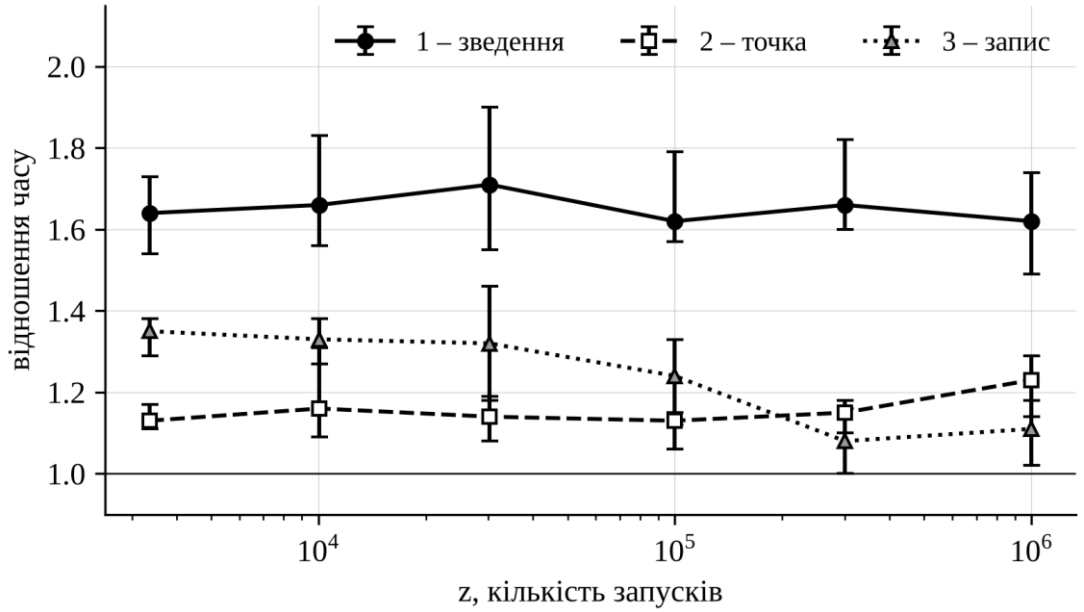


Рис. 1. Відношення часу виконання операції у схемі із сурогатними ключами до часу в каркасній схемі залежно від кількості запусків *z*: 1 – зведений запит, 2 – точковий запит, 3 – запит запису результатів запуску; відрізки – міжквартильний інтервал

Висновки. Побудовано каркасну БД конфігурацій і результатів запусків полігону, словниками якої є опції розрахункових програм, зокрема N і MU. Ключ кожного запуску складається з опцій розрахункової програми, тому запити і запис нових результатів виконуються в термінах предметної області без з'єднань і пошуку сурогатних ключів. У СКБД SQLite за однакових даних, індексів і фізичної організації каркасна схема виконує зведений запит у 1,6–1,7 раза, точковий у 1,1–1,2 раза, а запит запису результатів запуску в 1,1–1,35 раза швидше, ніж схема із сурогатними ключами. Найбільший і стабільний виграш має зведений запит, у якому впорядкування за каркасним ключем усуває групування через тимчасове B-дерево. Проте, відбувається збільшення файлу бази даний на 20 %. Предметом подальших досліджень є автоматизація програмування та використання таких застосунків на основі CASE-оболонки (платформи застосунків Low-Code) за допомогою агентного штучного інтелекту з використанням протоколу MCP [7].

Використання інструментів ШІ. ШІ-агент Claude (Anthropic, моделі Claude Fable 5.2 і Claude Opus 5.5) використано для редагування чернетки тез (основний розділ). Код, числові результати і бібліографічні записи створено та перевірено авторами.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

[1] Panchenko, B. E., Kovalev, Y. D., & Saiko, I. N. (2020). Numerical analysis of systems of singular integral equations of the first kind with an indefinable index in the problem of diffraction of plane waves on a rigid inclusion. *Cybernetics and Systems Analysis*, 56(4), 521–533. doi:10.1007/s10559-020-00268-z

[2] Wilkinson, M. D., Dumontier, M., Aalbersberg, I. J., Appleton, G., Axton, M., Baak, A., ... Mons, B. (2016). The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data*, 3, 160018. doi:10.1038/sdata.2016.18

[3] Gaffney, K. P., Prammer, M., Brasfield, L., Hipp, D. R., Kennedy, D., & Patel, J. M. (2022). SQLite: Past, present, and future. *Proceedings of the VLDB Endowment*, 15(12), 3535–3547. doi:10.14778/3554821.3554842

[4] Panchenko, B. E. (2012). Framework design of a domain-key schema of a relational database. *Cybernetics and Systems Analysis*, 48(3), 469–478. doi:10.1007/s10559-012-9426-7
[5] Panchenko, B. E., & Nazarenko, A. M. (2013). Framework analysis of the object domain of stationary dynamic problems of the elasticity theory for isotropic media with arbitrary discontinuities. *Cybernetics and Systems Analysis*, 49(1), 149–162. doi:10.1007/s10559-013-9495-2
[6] Panchenko, B. E., & Gajdabrus, V. N. (2013). A relational framework and case shells of a new type. *Cybernetics and Systems Analysis*, 49(3), 475–486. doi:10.1007/s10559-013-9532-1
[7] Anthropic. (2025). *Model Context Protocol specification*. Model Context Protocol. https://modelcontextprotocol.io/specification/2025-06-18

Pasenchenko T. O.

A carcass database schema for a mathematical physics boundary value problems training ground

Abstract. *A database of runs results for a mathematical physics boundary value problems training ground is built on the carcass (framework) data model. In SQLite it is compared with a surrogate-key schema under the same physical organization of the runs relation. The carcass schema executes a summary query 1.6–1.7 times faster, a point query 1.1–1.2 times faster and new run results insertion query 1.1–1.35 times faster, while its file is 20 % larger due to the wider key of the runs relation.*

Keywords: *database; numerical methods, carcass data model; SQLite; experiment; performance optimization; artificial intelligence.*