Міністерство освіти і науки України
Національний університет «Києво-Могилянська академія»
Факультет інформатики
Кафедра інформатики

Кваліфікаційна робота

освітній ступінь – бакалавр

на тему: **«РОЗРОБКА ЕТАЛОННОГО ТЕСТУ ТА ЕКСПЕРИМЕНТАЛЬНЕ ПОРІВНЯННЯ ГРАФОВИХ СКБД»**

Виконав: студент 4-го року навчання,
Спеціальності
122 Комп’ютерні науки
Корбут Даніїл Сергійович
Керівник Гулаєва Н.М.,_____
Рецензент _____
(прізвище та ініціали)

Кваліфікаційна робота захищена з оцінкою _____

Секретар ЕК _____

“___” _____ 20____ р.

Київ – 2024

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав. кафедри інформатики,
к.ф-м.н., доц. Гороховський С.С
_____
(підпис)
“_____” _____202_ р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на кваліфікаційну роботу

студенту Корбуту Даніілу факультету Інформатики 4 року навчання
ТЕМА **«РОЗРОБКА ЕТАЛОННОГО ТЕСТУ ТА ЕКСПЕРИМЕНТАЛЬНЕ ПОРІВНЯННЯ ГРАФОВИХ СКБД»**

Зміст текстової частини до кваліфікаційної:

Зміст
Анотація
Вступ
Розділ 1. Огляд еталонних тестів та інструментів тестування СКБД
Розділ 2. Розробка еталонного тесту та фреймворку для тестування графових СКБД
Розділ 3. Тестування графових СКБД
Висновки по роботі
Список літератури

Дата видачі “___” _____ 202_р.

Керівник
Гулаєва Н.М. _____
(підпис)

Завдання отримав
Корбут Д.С. _____
(підпис)

Тема: Розробка еталонного тесту та експериментальне порівняння графових СКБД

Календарний план виконання роботи:

№ п/п	Перелік робіт	Термін виконання	Примітка
1.	Отримання завдання на кваліфікаційну роботу	26.12.2022	
2.	Вивчення інформації за темою роботи	14.01.2024	
3.	Написання першого розділу	05.02.2024	
4.	Вибір інструментів для розробки програмної частини	10.02.2024	
5.	Створення програмної частини	15.03.2024	
6.	Написання другого розділу	30.03.2024	
7.	Проведення еталонного тестування	14.04.2024	
8.	Написання третього розділу	26.04.2024	
9.	Узгодження попередньої версії роботи з керівником	08.05.2024	
10.	Внесення змін до кваліфікаційної роботи відповідно до зауважень наукового керівника	11.05.2024	
11.	Попередній захист	14.05.2024	
12.	Корегування роботи за результатами попереднього захисту	21.05.2024	
13.	Захист кваліфікаційної роботи	30.05.2024	

Науковий керівник Гулаєва Н.М.

Виконавець кваліфікаційної роботи Корбут Д.С.

“___” _____

ЗМІСТ

СПИСОК СКОРОЧЕНЬ

СКБД – Система Керування Базами Даних

ОС – Операційна Система

TPC – Transaction Processing Performance Council

LDBC – Linked Data Benchmark Council

LPG – Labeled Property Graph

SQL – Structured Query Language

OLTP – Online Transaction Processing

OLAP – Online Analytical Processing

JDBC – Java Database Connectivity

ODBC – Open Database Connectivity

JVM – Java Virtual Machine

CQL – Cypher Query Language

nGQL – NebulaGraph Query Language

URL – Uniform Resource Locator

TPS – Transaction Per Second

NVMe – NVM Express

SSD – Solid State Drive

WSL – Windows Subsystem for Linux

ER – Entity Relationship

ECC – Error Correction Code

АНОТАЦІЯ

У роботі розглянуто еталонні тести та інструменти тестування СКБД. Проаналізовано наявні рішення. Запропоновано еталонний тест, що порівнюватиме продуктивність графових СКБД. Розроблено фреймворк для його реалізації. Проведено експериментальне тестування найпопулярніших графових СКБД з подальшим аналізом отриманих результатів.

Ключові слова: бази даних, графові СКБД, еталонне тестування, інструменти тестування

ВСТУП

Актуальність. Інтерес до графових даних постійно зростає, створюючи попит на засоби, що дозволять зберігати такі дані та ефективно доступатися до них. Такими засобами стали графові СКБД. Індустрія графових СКБД зародилася з появою першої комерційної графової системи Neo4j лише наприкінці 2000-х. Вона перебуває в постійному розвитку та потребує вивчення. Поява нових продуктів та оновлення існуючих потребують порівняння їх продуктивності. Порівняння відбувається за допомогою проведення еталонних тестів, які різноманітними метриками здатні оцінити можливості тієї чи іншої системи. Сфера графових СКБД за час свого прогресу встигла збагатитися певною кількістю еталонних тестів та інструментів тестування, проте їх різноманітність й рівень суттєво нижчий, ніж у СКБД з більш традиційними моделями даних.

Мета дослідження. Розробка еталонного тесту та інструментарію для оцінювання продуктивності графових СКБД.

Завдання дослідження. Оглянути та порівняти існуючі еталонні тести та інструменти тестування СКБД. Розробити власний еталонний тест для графових СКБД та фреймворк для його проведення. Оцінити продуктивність найпопулярніших графових СКБД за допомогою розробленого фреймворку.

Об’єкт дослідження. Еталонне тестування СКБД.

Предмет дослідження. Еталонні тести та інструменти для порівняння графових СКБД.

Джерела досліджень. Наукові публікації, документація програмного забезпечення, електронні ресурси, друкована література в електронних версіях.

Методи досліджень. Аналіз літератури та існуючих програмних засобів, обчислювальні експерименти.

У роботі здійснено огляд індустрії еталонного тестування СКБД, а також окремих еталонних тестів та інструментів тестування.

В ході роботи на основі оглянутих рішень запропоновано власний еталонний тест та розроблено фреймворк, що призначений для еталонного тестування СКБД з графовою моделлю даних. Предметною областю тесту є соціальна мережа, що складається з користувачів (вузлів) та дружби (зв’язків) між ними. Продуктивність конкретної СКБД перевіряється на основі виконання запитів, які шукають всіх друзів користувача на глибині N, що може бути налаштована.

Фреймворк надає можливість генерації тестового набору даних користувацького розміру та наступного завантаження його до цільової СКБД. Його основною функцією є еталонне тестування. Під час бенчмаркінгу інструмент збирає ряд метрик та формує з них підсумковий результат, що складається зі значень різних показників.

Робота складається з трьох розділів.

Перший розділ присвячено дослідженню розвитку сфери еталонного тестування СКБД, огляду існуючих еталонних тестів та інструментів тестування.

В другому розділі запропоновано власний еталонний тест, описано розроблений фреймворк для його проведення.

В третьому розділі проведено експериментальне еталонне тестування ряду СКБД за допомогою розробленого фреймворку, а також виконано аналіз отриманих результатів.

РОЗДІЛ 1. ОГЛЯД ЕТАЛОННИХ ТЕСТІВ ТА ІНСТРУМЕНТІВ ТЕСТУВАННЯ СКБД

1.1 Розвиток еталонних тестів

Перша потреба у розробці еталонних тестів виникла на початку 1970-х років. Тоді для впровадження централізованої системи обробки транзакцій у Bank of America виникла потреба у СКБД, що зможе обробляти 100 транзакцій за секунду. Відповідно, для перевірки придатності різних систем керування базами даних до таких навантажень мав бути розроблений бенчмарк або ж еталонний тест. Першою серед провідних компаній у сфері розробки СКБД на той час, що запропонувала своє рішення, була IBM та її тест TP1. Він був першою метрикою оцінки продуктивності СКБД, але мав серйозні недоліки: робота на стороні бази даних, відсутність врахування мережі, часу реакції оператору. В результаті послідовних операцій зі зняття та зарахування коштів на рахунок клієнта тест підраховував кількість транзакцій за секунду (TPS) [1].

З плином часу ринок комерційних СКБД невпинно ріс. Виробники систем звітували про неймовірні результати тесту TP1, що сягали декількох тисяч транзакцій на секунду. Проте через наведені вище недоліки тесту кінцеві користувачі систем не могли отримати навіть половини від такої продуктивності. У 1982 році Девід Девітт з Вісконсінського Університету створив більш детермінований еталонний тест, який покликаний був справедливо оцінювати продуктивність СКБД, проте його поява не змогла вирішити проблеми, а лише викликала ряд скандалів та конфліктів через незадовільні результати комерційного продукту від Oracle [1].

У 1985 році компанія Tandem запропонувала тест під назвою DebitCredit, який був орієнтований на роздрібно-банківські операції і вперше встановлював чіткі стандарти тестування. Цей тест відіграв важливу роль у процесі стандартизації тестування продуктивності СКБД. Однак, незалежним організаціям і користувачам все ще було потрібне об'єктивне й авторитетне мірило для оцінювання продуктивності [1].

В результаті у 1988 році було створено Раду з питань продуктивності обробки транзакцій (Transaction Processing Performance Council), до якої входила більшість великих виробників СКБД того часу. Мета TPC полягала у розробці низки стандартних тестів, результати яких обов'язково мали проходити аудит у відповідних організаціях. Першими затвердженими радою тестами стали TPC-A та TPC-B. Вони були досить добре прийняті індустрією. З появою нових потреб та запитів Рада затверджувала нові тести, що приходили на зміну застарілим та мали різну область застосування, але не завжди були вдалими. Серед них найбільш успішними (за частотою використання, кількістю результатів) вважаються TPC-C, TPC-E, TPC-H [1].

Наразі різноманітність можливих робочих навантажень, що користується попитом, є надзвичайно великою. СКБД можуть застосовуватися для обробки розподілених обчислень, мультимедійних даних, JSON, XML тощо. Для об'єктивного порівняння всього розмаїття потрібні окремі еталонні тести. Серед них YCSB, LinkBench, TeraSort, BIG-bench тощо.

Далі буде детальніше розглянуто кілька найпопулярніших еталонних тестів.

1.2 TPC-C

TPC-C є еталонним тестом онлайнової обробки транзакцій (OLTP) для реляційних СКБД, який було затверджено Радою з питань продуктивності обробки транзакцій (Transaction Processing Performance Council) у 1992. Він прийшов на зміну застарілим TPC-A та TPC-B [2].

На відміну від своїх попередників TPC-C має декілька типів транзакцій, більш складну схему бази даних та загальну структуру виконання. Як OLTP-тест, він симулює роботу повноцінного середовища, у якому оператори терміналів бази даних виконують транзакції до системи. Варто зазначити, що TPC-C не націлений на діяльність конкретного сегменту бізнесу, а є представленням будь-якої сфери продажу та розповсюдження товарів і послуг [2].

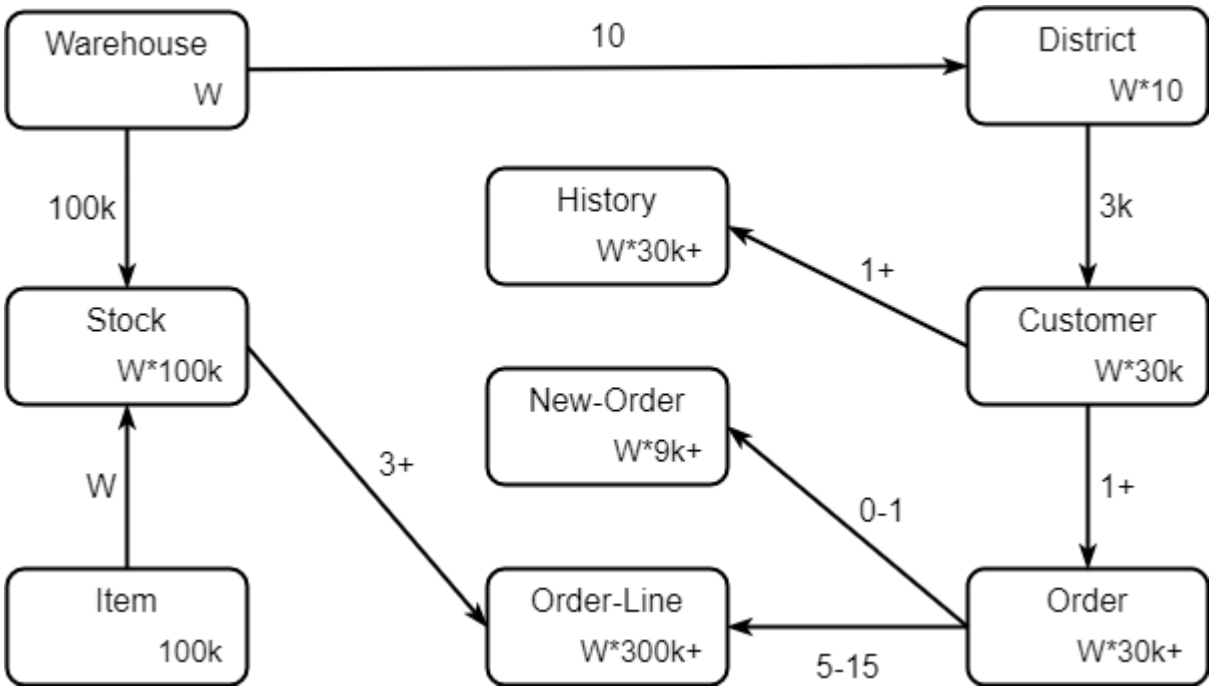

Рисунок 1.1 ER-діаграма специфікації TPC-C [32]

Предметна область тесту – моделювання роботи оптового постачальника деталей, що працює з певною кількістю складів. ER-діаграма наведена на рисунку 1.1. Кількість складів масштабується у ході тестування пропорційно до обчислювальних можливостей системи. Масштабування відбувається, якщо досягається межа продуктивності, та продовжується доки час відповіді 90% транзакцій не перевищує 5 секунд. При цьому кожен склад повинен обслуговувати десять торгових районів, а кожен торговий район – три тисячі клієнтів. Торговий район може провести не більше 1.2 замовлень за хвилину (чим і зумовлена межа продуктивності). Оператор терміналу у своєму торговому районі має можливість в будь-який час здійснити будь-яку операцію з п’яти видів, що надаються системою введення замовлень. Їх частота й зміст базується на реальних сценаріях використання [2].

Найбільш частою операцією є додавання нового замовлення, що в середньому складається з десяти позицій. Кожен склад намагається підтримувати наявність всього каталогу розміром у 100 000 позицій й виконувати замовлення на основі цих запасів. Проте, моделюючи реальні умови, TPC-C вимагає, щоб близько 10% замовлень постачались з інших складів. Наступною частою операцією є запис платежу клієнта. Рідше

оператори запитують статус оформлених замовлень, перевіряють рівень запасів на своєму складі, щоб дізнатися про можливий дефіцит, а також обробляють партію з десяти замовлень на доставку. Це і формує п’ять видів операцій [2].

Результатом еталонного тесту TPC-C є ряд показників:

- Пропускна здатність – кількість замовлень, що можуть бути повністю оброблені за хвилину. Виражається у tpm-C.
- Співвідношення ціни до пропускної здатності – показник, що обраховується діленням ціни системи на отриману пропускну здатність. При цьому ціна формується не лише з вартості машини, на якій проводиться тест, а й з вартості додаткового обладнання, необхідного програмного забезпечення та трирічних витрат на обслуговування. Виражається у одиниці ціна/tpm-C.
- Співвідношення енергоспоживання до пропускної здатності – метрика, що вираховує енергоспоживання у ватах на тисячу tpm-C, використовуючи складну методику розрахунку енергоспоживання. Є опціональною та майже не зустрічається у опублікованих результатах. Вимірюється у одиниці ват/Ktpm-C.

Фактично, основною збираною метрикою є пропускна здатність. Інші показники вираховуються вручну, спираючись на неї.

Еталонний тест має чітку специфікацію, що містить зразки коду для реалізації та велику кількістю вимог, але не надає готового інструментарію для його проведення. Це завдання покладається на тих, хто проводитиме тестування.

TPC-C є загальновизнаним еталонним тестом для OLTP-навантажень та був використаний для перевірки багатьох СКБД, але не є підходящим для OLAP-систем, потреба в яких сильно росла з плином часу.

1.3 Yahoo! Cloud Serving Benchmark

Yahoo! Cloud Serving Benchmark (YCSB) – еталонний тест, розроблений у 2010 році компанією Yahoo. Здебільшого відомий у області тестування NoSQL баз даних. YCSB найчастіше використовується для порівняння продуктивності

СКБД, що мають багато вузлів в публічній хмарі або іншій розподіленій інфраструктурі [3].

Завдяки свої модульності та використанню JDBC Yahoo! Cloud Serving Benchmark підтримує велику кількість як NoSQL, так і SQL систем керування базами даних. Підтримуються моделі даних ключ-значення, документо-орієнтована, родини стовпців та реляційна. [3].

YCSB надає користувачам робочі навантаження для імітації різних реальних сценаріїв. Конкретної предметної області тест не має. Робочі навантаження зазвичай складаються з комбінації операцій читання, запису, оновлення та видалення, з різним співвідношенням залежно від конкретного варіанту використання [4]. YCSB надає 6 попередньо визначених робочих навантажень:

- Інтенсивне оновлення – поєднання операцій запису й читання у рівному співвідношенні.
- Переважне читання – навантаження, що складається з 95% операцій читання, решта – операції запису.
- Лише читання – складається лише з операцій читання.
- Читання останніх – навантаження, в якому відбувається додавання нових записів, але найчастіше зчитуються лише останні з них.
- Короткі діапазони записів – замість отримання окремих записів, відбуваються запити невеликих діапазонів записів.
- Читання-зміна-запис – відбувається читання запису, його модифікація користувачем та збереження.

До того ж користувачі можуть визначати власні робочі навантаження, адаптовані до їхніх особливих вимог [5].

Для кожного виду робочого навантаження YCSB збирає наступні метрики:

- Пропускна здатність – кількість операцій на секунду (ops/sec).
- Середня, мінімальна, максимальна затримки (мілісекунди)

- 95-й та 99-й перцентилі затримки (мілісекунди)
- Загальний час виконання (мілісекунди)

Затримка є часом виконання окремого запиту.

Перед запуском бенчмарку потрібно завантажити дані в базу, що потребує тестування. YCSB надає інструменти для створення штучних наборів даних різного розміру, які можна завантажити в систему за допомогою клієнта YCSB [6].

Після завантаження даних можливо запустити еталонний тест, запустивши клієнт YCSB з потрібною конфігурацією робочого навантаження. Клієнт генеруватиме безперервний потік операцій на основі обраної специфікації робочого навантаження і надсилатиме запити до бази даних [6].

1.4 LinkBench

Еталонний тест під назвою LinkBench був розроблений компанією Facebook (нині Meta) у 2013 році. Він створювався задля перевірки, яка база даних є найбільш ефективною для зберігання даних соціальної мережі Facebook [7].

id	int64	unique identifier
type	int32	type of object
version	int64	tracks version of object
update_time	int32	last modification (UNIX timestamp)
data	text	data payload

(a) Object (graph node). id is unique key.

id1, id2	int64	ids of edge's endpoints
atype	int64	type of the association
visibility	int8	visibility mode of the edge
timestamp	int32	a client-provided sort key
data	varchar	small additional data payload

(b) Association (graph edge). (id1, atype, id2) is unique key.

Рисунок 1.2 Схема соціального графу LinkBench [7]

У випадку з Facebook дані зберігались у вигляді соціального графу, де об'єкти або ж вузли, такі як сторінки, люди, коментарі, поєднуються зв'язками, які відображають різні види відносин між вузлами. Схема наведена на рисунку 1.2. Типовими прикладами таких відносин у соціальній мережі є дружба між користувачами, вподобання певного посту або право на його власність тощо.

Вузли та зв’язки містять певні метадані та можуть містити корисне навантажене довільного характеру. Дані у Facebook, використовуючи таку модель, зберігались в базах даних MySQL. Отож метою LinkBench є емуляція реального робочого навантаження бази даних, яка зберігає дані у вигляді соціального графу, а також реалістична оцінка продуктивності цієї бази. Для проведення тесту LinkBench генерує штучний набір даних подібний до справжнього, забезпечуючи його ключові особливості [7].

Метрики, що збирає LinkBench під час еталонного тестування такі:

- Пропускна здатність – кількість запитів за секунду (requests/second)
- Загальний час виконання
- Максимальна та середня затримка (мілісекунди)
- 25-й, 50-й, 75-й, 95-й та 99-й перцентилі затримки

Затримка вимірюється часом, що проходить від виконання операції клієнтом до отримання всіх результатів, якщо йдеться про операції читання, або доки не буде отримано підтвердження, що операцію запису завершено. Для всього тесту обчислюється пропускна здатність, а для кожного виду операцій надається окрема статистика затримок [7].

Головним програмним компонентом LinkBench є драйвер, який імітує клієнта бази, що тестується. За підключення до бази даних відповідає інтерфейс Graph Store Adapter. Використовуючи відповідну реалізацію цього інтерфейсу, можна тестувати будь-яку базу даних, яка підтримує операції, що вимагає тест. Завдяки цьому LinkBench підтримує велику кількість СКБД з різними моделями даних: реляційні (MySQL, PostgreSQL), документо-орієнтовані (MongoDB), ключ-значення (RocksDB) та родини стовпців (HBase) [7].

Окрім Graph Store Adapter, драйвер містить дві важливі складові: Генератор Графу (Graph Generator) та Генератор Робочого Навантаження (Workload Generator). Вони використовуються в ході двох фаз тестування [7].

Під час першої фази відбувається генерація соціального графу за допомогою Graph Generator. Потім згенеровані дані завантажуються у базу даних, що тестується [7].

Друга фаза, що безпосередньо відповідає за еталонне тестування, називається фазою запитів. Під час цієї фази драйвер створює багато потоків, які починають відправляти паралельні запити до бази даних. Зміст цих запитів визначає Workload Generator. Він генерує набори операцій, які імітують реальне робоче навантаження соціальної мережі [7].

1.5 Інструменти тестування СКБД

Проведення правильного еталонного тестування СКБД потребує кропіткої роботи з налаштування середовища та підготовки, що робить таке тестування важким з практичної точки зору. Інструменти тестування СКБД покликані спростити цю задачу. Вони надають можливість проводити різноманітні тести готовими програмними засобами, збираючи більш прості показники, водночас роблять це достатньо надійно, щоб зберігалась доцільність у порівнянні результатів. Також інструменти тестування часто полегшують підготовчі етапи бенчмаркінгу, що пов’язані з генерацією даних, створенням схеми та завантаженням набору даних у базу.

Назва	Тип продукту	Платформа	СКБД	Характеристика
BenchBase	Відкритий	Windows, Linux	Реляційні з підтримкою JDBC	Широкий список популярних тестів (TPC-C, TPC-H, YCSB та ін.), можливість додавання власних
Benchmark Factory for Databases	Пропрієтарний	Windows	Oracle Database, SQL Server, DB2, SAP, MySQL	Графічний інтерфейс, наявність найпоширеніших тестів від TPC (TPC-H, TPC-C, TPC-D, TPC-E), можливість запису та відтворення реального навантаження
sysbench	Відкритий	Linux	MySQL, PostgreSQL	Колекція OLTP-тестів та тестів на швидкодію оперативної пам'яті, процесора, файлової системи від розробників інструменту. Можливість додавання власних
OSDLDBT	Відкритий	Linux	MySQL, PostgreSQL, SQLite, CockroachDB	Набір тестів, що є авторською імплементацією TPC-C, TPC-H, TPC-E, TPC-DS
HammerDB	Відкритий	Windows, Linux	Oracle Database, Microsoft SQL Server, IBM Db2, MySQL, MariaDB, PostgreSQL	Графічний інтерфейс, модифіковані реалізації тестів TPC-C та TPC-H. Наявність порталу для обміну результатами тестування.
pgbench	Відкритий	Windows, Linux	PostgreSQL	Є стандартним компонентом PostgreSQL, відтворює навантаження, подібні до тесту TPC-B або користувацькі сценарії.
tpce-mysql	Відкритий	Linux	MySQL	Відповідно до назви, реалізує тест на основі специфікації TPC-E для СКБД MySQL

Таблиця 1.1 Порівняння інструментів тестування СКБД

[8][9][10][11][12][13][14]

В таблиці 1.1 наведено загальне порівняння більшості доступних інструментів для тестування систем керування базами даних за рядом властивостей. З цього порівняння можна зробити декілька висновків. По-перше, інструменти тестування СКБД мають здебільшого відкритий вихідний код. По-друге, за виключенням комерційного продукту Benchmark Factory for Databases, вони є або кросплатформними, або підтримують лише Linux. Враховуючи, що у виробничому середовищі СКБД частіше працюють на машинах під управлінням Linux, такий результат є передбачуваним. Найбільш підтримуваними СКБД у інструментах є MySQL та PostgreSQL.

Перейдемо до детальнішого розгляду декількох популярних представників таких засобів. Їх було обрано з огляду на кількість підтримуваних ними СКБД, розмаїття доступних до використання тестів та наявність характерних особливостей.

1.6 HammerDB

HammerDB – програмне забезпечення, призначене для навантажувального та еталонного тестування ряду СКБД, що має відкритий вихідний код. Розроблене Стівом Шоу, експертом у сфері баз даних, за підтримки TPC. Мета цього інструменту полягає у полегшенні та автоматизації процесу тестування СКБД у різних середовищах.

Типи підтримуваних СКБД	Реляційні
Підтримувані СКБД	Oracle Database, Microsoft SQL Server, IBM Db2, MySQL, MariaDB та PostgreSQL
ОС	Windows, Linux
Графічний інтерфейс	Так
Інтерфейс командного рядку	Так
Еталонні тести	TPROC-C, TPROC-H

Таблиця 1.2 Інформація про HammerDB [12]

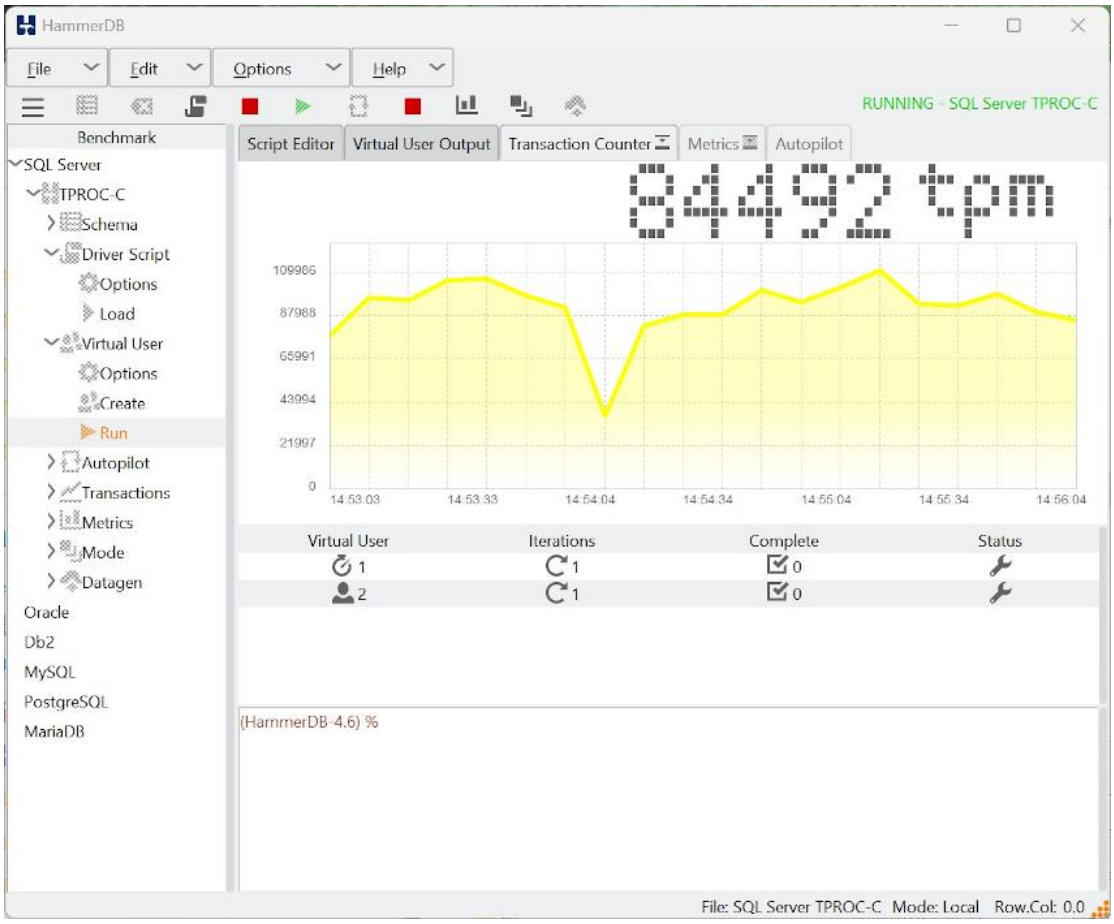

Рисунок 1.3 Графічний інтерфейс HammerDB

Графічний інтерфейс полегшує процес налаштування. Також є можливості з візуалізації результатів тестування у реальному часі (рисунок 1.3) [12].

HammerDB надає можливості з генерації наборів даних, їх зручного завантаження до цільової бази даних, створення її схеми та безпосереднього проведення тесту. Інструмент симулює робоче навантаження на СКБД, передбачене еталонними тестами, відправляючи відповідні запити багатьма віртуальними користувачами. Кількість таких користувачів довільно налаштовується. Зміна їх кількості дозволяє перевіряти масштабованість системи [12].

Список доступних еталонних тестів, що дозволяє виконати HammerDB, обмежується двома: TPROC-C та TPROC-H.

TPROC-C – це OLTP-тест, створений на основі специфікації TPC-C з модифікаціями, які спрощують роботу з HammerDB в підтримуваних

середовищах баз даних. Вибір на користь цієї специфікації було зроблено з огляду на визнання тесту та авторитетність організації TPC в індустрії еталонних тестів [12].

TPROC-H є тестом для OLAP-систем. Він також є модифікацією стандарту визнаного еталонного тесту TPC-H від TPC [12].

Оскільки обидва еталонні тести відрізняються від стандартної специфікації тестів TPC, то порівняння результатів, що були отримані за допомогою HammerDB, з опублікованими результатами тестів TPC, не є доцільним. Водночас на сайті інструменту можна обмінюватися результатами тестувань. Їх порівняння зі свого боку має деякий сенс, враховуючи проведення тесту однаковим інструментом.

Під час тестування HammerDB збирає показники транзакцій за хвилину та навантаження на процесор [12].

1.7 BenchBase

BenchBase – фреймворк для еталонного тестування реляційних СКБД з відкритим вихідним кодом. Є офіційною модифікацією та продовженням відомого інструменту OLTPBench. Його було розроблено групою вчених, що є спеціалістами в області СКБД [8].

Типи підтримуваних СКБД	Реляційні
Підтримувані СКБД	MySQL, PostgreSQL, Oracle Database, Microsoft SQL Server, MySQL, SQLite, DB2 та інші з підтримкою JDBC
ОС	Windows, Linux
Графічний інтерфейс	Ні
Інтерфейс командного рядку	Так
Еталонні тести	AuctionMark, CH-benCHmark, Epinions.com, NoOp, OT-Metrics, Resource Stresser, SEATS, SIBench, SmallBank, TATP, TPC-C, TPC-H, Twitter, Voter, Wikipedia, YCSB

Таблиця 1.3 Інформація про BenchBase [8]

На відміну від HammerDB цей інструмент не надає графічного інтерфейсу. Натомість має функціональний інтерфейс командного рядка.

BenchBase може здійснювати еталонне тестування будь-якої реляційної бази, що підтримує Java Database Connectivity (JDBC). Отже, можливе

використання на більшості реляційних СКБД. Також інструмент надає можливості з перетворення запитів у потрібний цільовій СКБД діалект для забезпечення сумісності [8].

BenchBase має вражаючу різноманітність тестів, що постачаються з фреймворком. Наприклад, до переліку входить ряд провідних тестів TPC-C, TPC-H, YCSB, LinkBench (підтримується тільки в OLTPBench), що були згадані у попередніх підрозділах. Також BenchBase має цікаві тести, що базуються на робочих навантаженнях та схемах соціальної мережі Twitter або відомої онлайн-енциклопедії Wikipedia. Повний перелік, що надає фреймворк, складається з понад 10-ти бенчмарків. Таким чином, встановлення й налаштування лиш одного інструменту запобігає витраті часу на встановлення кожного з тестів окремо [8].

Кожен доступний еталонний тест піддається гнучкому налаштуванню за допомогою відповідних конфігураційних файлів.

Результатом тестування є час виконання тесту, пропускна здатність (вимірюється у транзакціях за секунду) та затримки запитів до СКБД (повна статистика, мінімальна, максимальна, середня, а також 25-й, 50-й, 75-й, 90-й, 95-й, 99-й перцентилі) [8].

1.8 Benchmark Factory for Databases

Benchmark Factory for Databases – пропрієтарний інструмент для тестування СКБД, що розроблено компанією Quest Software.

Типи підтримуваних СКБД	Реляційні
Підтримувані СКБД	Oracle Database, SQL Server, DB2, SAP, MySQL та інші з підтримкою ODBC
ОС	Windows
Графічний інтерфейс	Так
Інтерфейс командного рядку	Ні
Еталонні тести	TPC-H, TPC-C, TPC-D, TPC-E, ASP3AP

Таблиця 1.4 Інформація про Benchmark Factory for Databases [9]

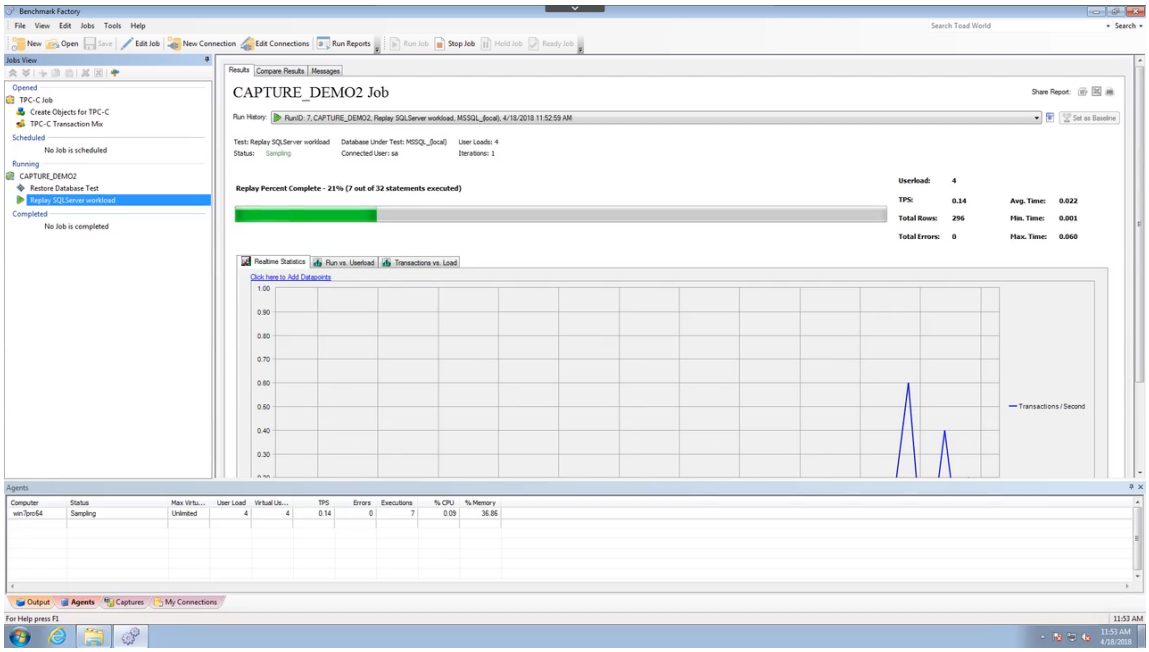

Рисунок 1.4 Графічний інтерфейс Benchmark Factory For Databases

Стандартний набір еталонних тестів складається з тестів на масштабованість, а також декількох представників від TPC: TPC-H, TPC-C, TPC-D, TPC-E. Проте особливістю цього продукту є можливість створювати власні тести на основі реального навантаження. Програма записує транзакції, що надходять до СКБД, впродовж визначеного часу й дозволяє відтворювати це навантаження на різних конфігураціях. Це дозволяє проводити тести з урахуванням індивідуальних особливостей навантаження системи клієнта [9].

Benchmark Factory під час тестування збирає дані про кількість транзакцій на секунду та затримку (мінімальну, максимальну, середню). Зібрана інформація візуалізується на графіках у реальному часі (рисунок 1.4) [9].

1.9 Еталонні тести та інструменти тестування графових СКБД

Індустрія графових СКБД є доволі молодою на відміну від індустрії реляційних, яка бере свій початок ще у двадцятому столітті. З цієї причини еталонне тестування графових баз даних ще не досягло рівня реляційних СКБД та інших систем з усталеними моделями даних. Однак, індустрія не стоїть на місці та працює над новими рішеннями у цій сфері.

Як було зазначено раніше, важливо мати організацію, що відповідатиме за розробку надійних тестів, які матимуть чітку специфікацію, а також

обов’язковий аудит результатів. У сфері графових даних цю місію виконує некомерційна Рада з питань еталонного тестування зв’язаних даних (Linked Data Benchmark Council). До її складу входять представники індустрії та науки. Для графових СКБД типу LPG було розроблено два еталонні тести: Social Network Benchmark (SNB) та Financial Benchmark (FinBench).

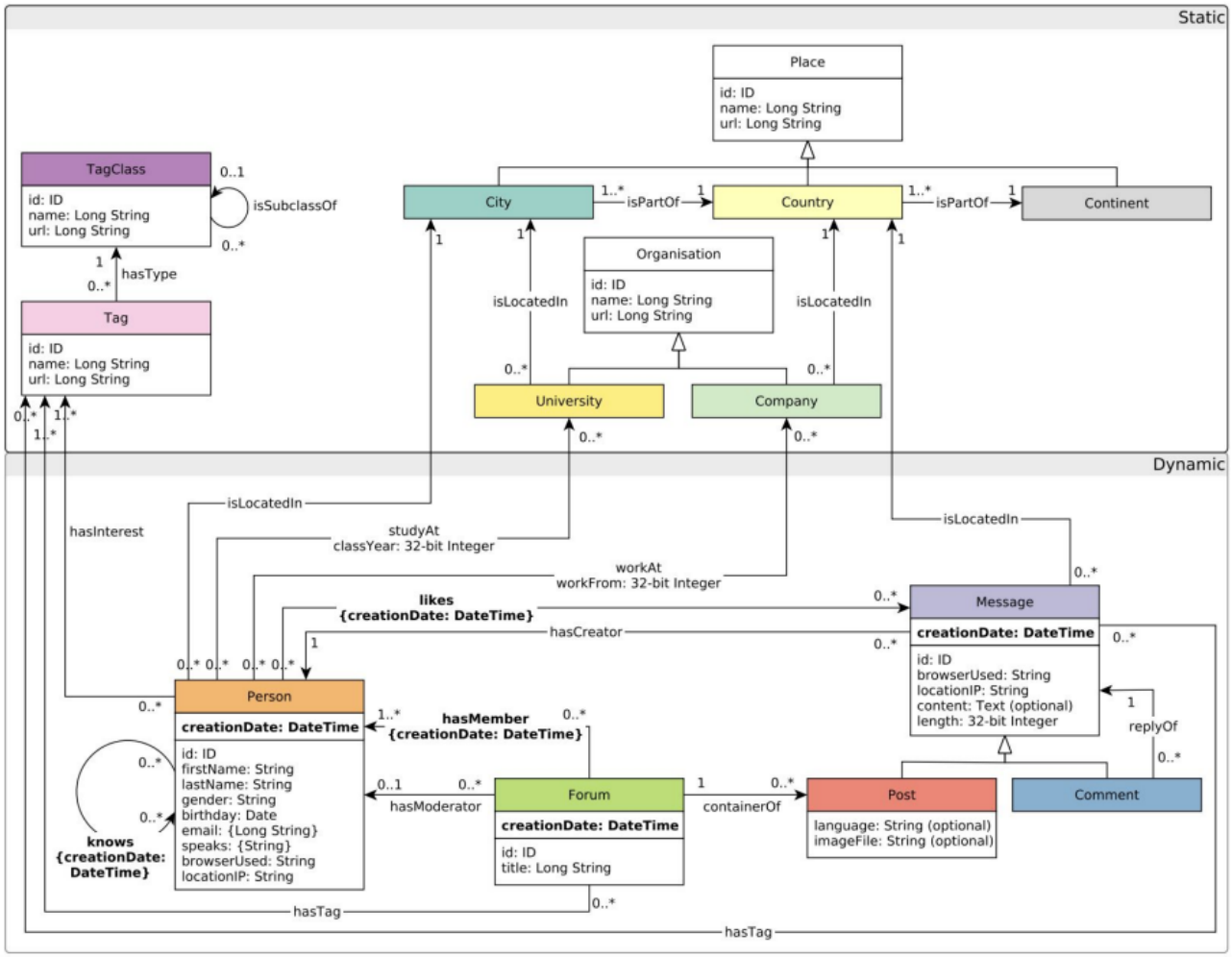

Рисунок 1.5 Схема графу Social Network Benchmark від LDBC [27]

Відповідно до назви, предметною областю Social Network Benchmark є соціальна мережа. Схема графу наведена на рисунку 1.5 [27].

	Business Intelligence v1.0	Interactive v1.0	Interactive v2.0
focus	OLAP	OLTP	OLTP
typical queries	multi-hop / path / subgraph queries with filtering & aggregation	2-3 hop top-k queries with filtering	2-3 hop top-k queries with filtering
data generator	Spark Datagen	Hadoop Datagen	Spark Datagen
refresh operations	inserts and deletes	inserts	inserts and deletes
target metric	throughput score power score	throughput (ops/s)	throughput (ops/s)
largest SF	10 000+	1 000	10 000+

Рисунок 1.6 Порівняння робочих навантажень SNB [27]

Тест передбачає два робочих навантаження: Business Intelligence та Interactive. Навантаження Business Intelligence зосереджене на складних запитах з агрегуванням і об'єднанням, використовуючи велику частину графу, а також невеликими пакетними операціями вставки/видалення. Навантаження Interactive складається зі складних запитів на читання, що отримують доступ до сусідів певної вершини графа, а також з операцій оновлення, які постійно додають до графу нові дані. Порівняння цих навантажень наведене на рисунку 1.6. На ньому видно типові запити, метрики, що збираються, та інші особливості [27].

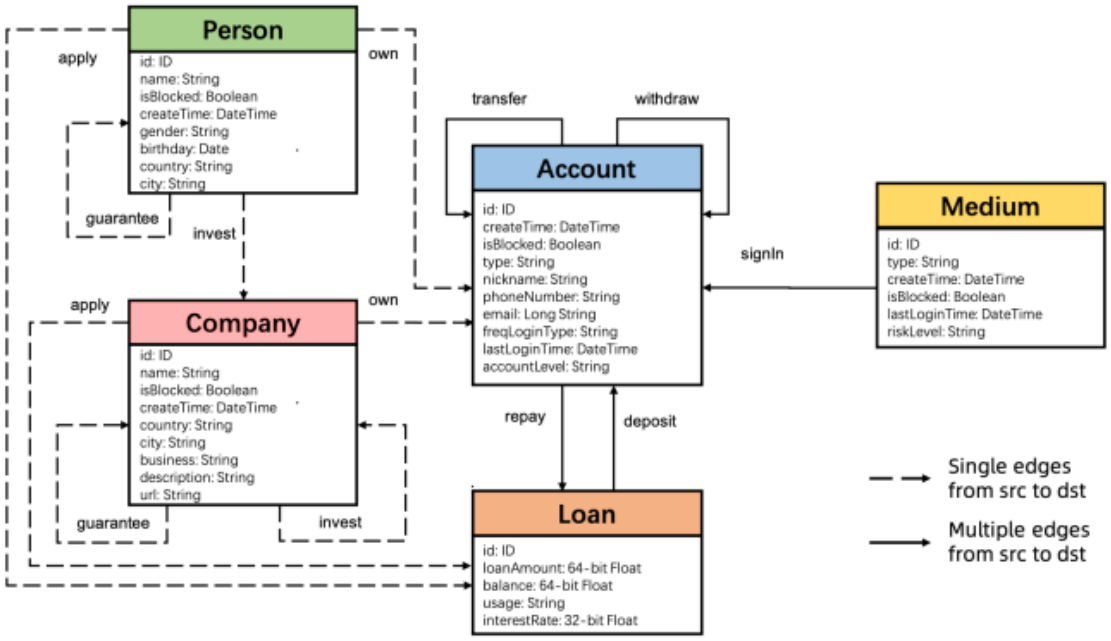

Рисунок 1.7 Схема графу Financial Benchmark від LDBC [28]

Тест FinBench оперує у предметній області фінансової системи. Він націлений на оцінку продуктивності графових СКБД в фінансових сценаріях (запобігання шахрайству, контроль ризиків тощо). Схема даних наведена на рисунку 1.7 [28].

Робоче навантаження тесту складається з наступних видів запитів:

- 12 складних зчитувань: зіставлення точних шаблонів, включаючи цикли та дерева, що починаються з однієї або двох вершин.
- 6 простих зчитувань: виявлення сусідів вершини типу Account.
- 19 запитів на запис: вставки, оновлення, видалення.
- 3 запити на читання-запис.

Результатом тесту є показник пропускної здатності (кількість операцій за секунду) [28].

Враховуючи складність коректного проведення цих тестів та потребу в аудиті, на сайті LDBC розміщено менше 10 результатів для SNB та жодного результату для FinBench (це також пояснюється публікацією тесту лише у 2023 році).

Назва	Тип продукту	Платформа	СКБД	Характеристика
graphdb-benchmarks	Відкритий	Windows, Linux	Titan, OrientDB, Neo4j, Sparksee	Складається з чотирьох робочих навантажень: кластеризація, масова вставка, одиночна вставка та запитів на пошук сусідніх вершин, найкоротшого шляху
GDB Test-suite	Відкритий	Windows, Linux	ArangoDB, BlazeGraph, Neo4j, OrientDB, Sparksee, SQLG, Titan	Перевіряє продуктивність за допомогою примітивних запитів, що являють собою CRUD-операції
Benchgraph	Відкритий	Windows, Linux	Memgraph, Neo4j	Реалізує комплексні робочі навантаження на базі тесту SNB від LDBC та власні на наборі даних соц. мережі Pokec. Інструмент від розробників Memgraph
WDBench	Відкритий	Windows, Linux	Apache Jena, Virtuoso, Blazegraph, Neo4j	Складається з робочих навантажень сервісу Wikidata та орієнтований на системи з підтримкою SPARQL, але має адаптовані версії більшості запитів для Neo4j

Таблиця 1.5 Порівняння інструментів тестування графових СКБД [29][30][31][33]

Існуючі рішення для тестування графових СКБД характеризуються доволі обмеженою підтримкою суто графових систем та високою складністю використання. Порівняння існуючих інструментів тестування наведено в таблиці 1.5.

Еталонні тести та інструменти тестування є різноманітними з точки зору предметної області або пропонованих робочих навантажень, але найчастіше збирають схожі метрики: кількість транзакцій за одиницю часу, затримки окремих запитів, на основі яких обчислюються середні, крайні показники та

значення розповсюджених перцентилів. Еталонне тестування графових СКБД наразі обмежується складністю доступних інструментів та недостатньою підтримкою ними популярних СКБД. З цієї причини завданням роботи є створення простішого у застосуванні тесту з можливістю додавання підтримки нових СКБД.

РОЗДІЛ 2. РОЗРОБКА ЕТАЛОННОГО ТЕСТУ ТА ФРЕЙМВОРКУ ДЛЯ ТЕСТУВАННЯ ГРАФОВИХ СКБД

2.1 Еталонний тест для графових СКБД

Ключовою особливістю графових СКБД є спосіб зберігання даних та запитів до них. На відміну від реляційної моделі, де дані зберігаються в окремих таблицях та пов’язані за допомогою ключів, графова модель являє собою абсолютно інший підхід, заснований на принципах теорії графів. Модель складається з вершин (або вузлів) та ребер (або зв’язків), що можуть їх поєднувати. Залежно від особливостей конкретних СКБД вершини та ребра можуть зберігати різні властивості та ідентифікатори, що й формують набір даних в базі. Використання графової моделі надає користувачу переваги графової теорії та дозволяє послуговуватися різними її алгоритмами, як-от пошук найкоротшого шляху алгоритмом Дейкстри, пошук у глибину, пошук у ширину тощо.

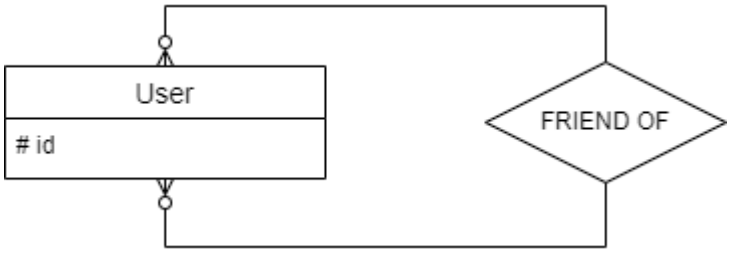

Рисунок 2.1 ER-модель

Класичним практичним прикладом влучного застосування графової системи керування базами даних є соціальна мережа: список користувачів, що можуть бути друзями один з одним (рисунок 2.1).

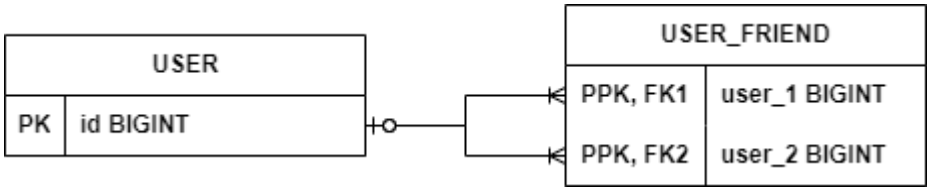

Рисунок 2.2 Реляційна модель

В реляційній моделі таку мережу можна представити у вигляді двох таблиць: одна, що містить користувачів та інформацію про них, інша, що

зберігає дані про дружбу між користувачами, поєднуючи їх ідентифікатори у окремих записах (рисунок 2.2).

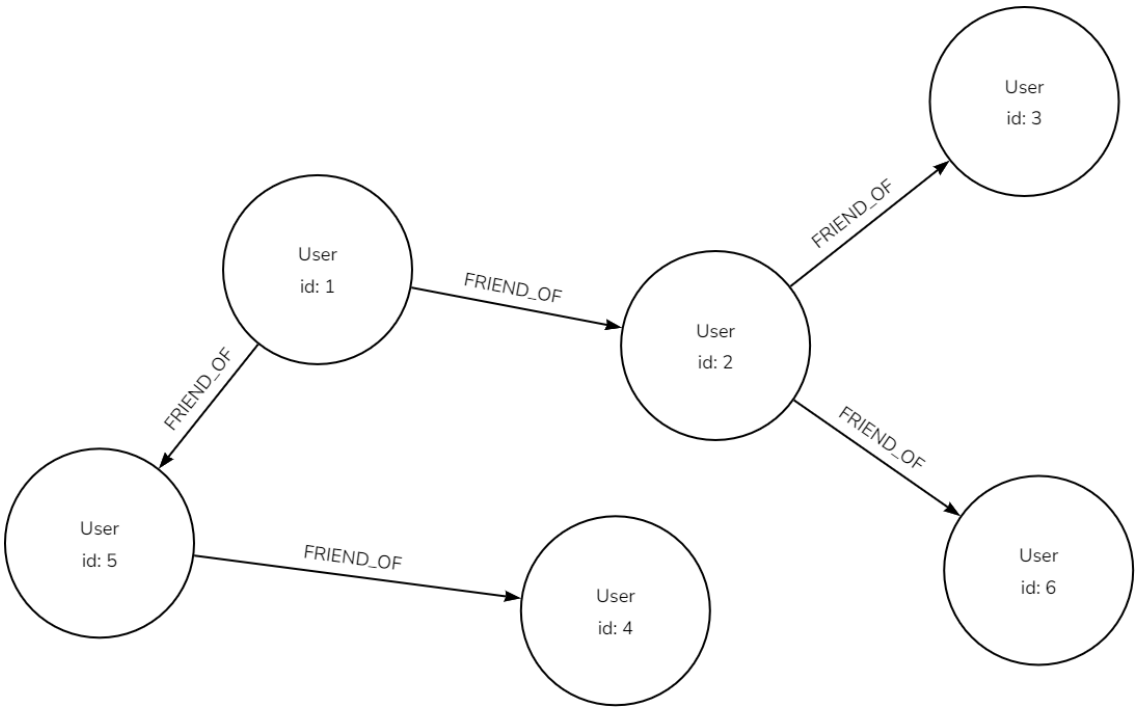

Рисунок 2.3 Графова модель

У графовій ж моделі соціальну мережі користувачів можна представити у вигляді вузлів, а дружбу між ними – зв’язками між відповідними вузлами (рисунок 2.3).

Пошук безпосередніх друзів користувача є простим завданням для реляційних СКБД, адже потребує лише фільтрації потрібних записів з таблиці, що зберігає інформацію про дружбу між користувачами. Проте відомі соціальні мережі часто мають функціонал, що рекомендує користувачу можливих друзів. Для формування цього списку виконується пошук кандидатів серед друзів безпосередніх друзів користувача. Він може сягати великої глибини.

Depth	Execution time (seconds) for 1 million users	Count result
2	0.016	~2,500
3	30.267	~125,000
4	1,543.505	~600,000
5	Not finished	—

Рисунок 2.4 Час виконання запитів на MySQL [15]

Depth	Execution time (seconds) for 1 million users	Count result
2	0.01	~2,500
3	0.168	~110,000
4	1.359	~600,000
5	2.132	~800,000

Рисунок 2.5 Час виконання запитів на Neo4j [15]

Перевага графової СКБД над реляційною в контексті моделі соціальної мережі та пошуку друзів до певної глибини гарно продемонстрована у книзі Алекси Вукотича та Нікі Ватта «Neo4j in Action» [15]. В ній експериментально порівняно швидкість реляційної СКБД MySQL з графовою Neo4j. Збільшення глибини дуже негативно впливає на продуктивність для реляційної СКБД (рисунок 2.4), проте мало змінює час виконання еквівалентного запиту у графовій Neo4j (рисунок 2.5). Непридатність реляційної моделі для виконання такого завдання пояснюється тим, що для отримання всіх друзів на глибині 5 необхідно з’єднувати (JOIN) таблицю, що містить зв’язки між користувачами, 5 разів, відповідно. Це призведе до створення величезної кількості надлишкових даних, обробка яких потребуватиме великих обчислювальних потужностей та багато часу [15].

Отже, цілком очевидно, що графові СКБД значно виграють реляційним у задачі пошуку друзів глибини N, проте постає питання, яка з графових СКБД є більш продуктивною у цьому. Щоб отримати відповідь, потрібно створити відповідний еталонний тест, що зможе надійно порівнювати продуктивність різних графових систем керування базами даних.

Пропонований еталонний тест відправляє деяку кількість запитів до цільової СКБД та отримує результат їх виконання. Окремий запит шукає всіх друзів окремого користувача до глибини N. Ідентифікатор цього користувача (id) обирається випадково в межах доступного діапазону. Глибина обмежує кількість переходів по ребрам графу від початкової вершини. Від неї залежить обсяг роботи з обходу графа, що має виконати СКБД, а отже й час виконання

запиту. Щоб не зчитувати всіх знайдених користувачів, запит повертає їх кількість (COUNT), максимально зменшуючи довжину відповіді. При обході графа один користувач може потрапляти в результат декілька разів, тому передбачається пошук лише унікальних користувачів (DISTINCT).

```
MATCH (:User {id: 10})-[:FRIEND_OF*..5]-(b)
RETURN COUNT(DISTINCT b) AS result
```

Рисунок 2.6 Приклад CQL-запиту еталонного тесту

На рисунку 2.6 наведено приклад типового запиту еталонного тесту мовою CQL, що шукає кількість друзів користувача з ідентифікатором 10 до глибини 5.

2.2 Фреймворк для еталонного тестування

Для реалізації еталонного тесту буде розроблено фреймворк, що повинен задовольняти ряд вимог.

Основний функціонал:

1. Генерація набору даних.
    - Можливість генерації штучного набору даних довільного розміру, що включатиме в себе користувачів та дружбу між ними.
    - Налаштування допустимої кількості друзів у користувачів.
2. Завантаження набору даних.
    - Можливість завантажити набір даних до цільової СКБД у автоматичному режимі.
    - Опціональна очистка СКБД перед завантаженням.
3. Еталонне тестування.
    - Відправка запитів, кількість яких задається користувачем, до цільової СКБД.
    - Збір та надання метрик.
    - Налаштування максимальної глибини пошуку.
    - Можливість відправки запитів у декілька потоків, кількість яких задається користувачем.

Однією з найважливіших частин тестування є метрики. Саме на їх основі відбувається оцінка та порівняння продуктивності СКБД. На основі проаналізованих у попередніх розділах еталонних тестів та інструментів тестування було відібрано найбільш поширені та репрезентативні показники, що формуватимуть результат тестування.

По-перше, фреймворк має збирати інформацію про затримку кожного запиту. Далі обчислити та надати користувачу наступні метрики:

- Максимальна, мінімальна та середня затримки – дозволяють оцінити крайні та середні випадки.
- 25-й, 50-й, 75-й, 95-й та 99-й перцентилі затримки – дозволяють побачити загальну картину, відсікаючи пікові точки на великих перцентилях.

Окрім затримок кожного запиту, фреймворк під час еталонного тестування повинен забезпечити підрахунок таких показників:

- Загальний час виконання тесту у секундах.
- Кількість транзакцій за секунду (TPS) на основі загальної кількості запитів та часу їх виконання. Однією транзакцією вважається відправка запиту до СКБД та отримання відповіді. Фактично, TPS є головним показником для порівняння продуктивності СКБД.
- Кількість успішних та невдалих запитів та їх співвідношення. Невдалим вважається запит, що повернув помилку замість очікуваного результату. За допомогою відсотку успішних запитів можна оцінити надійність цільової бази даних.

2.3 Інструменти розробки

Вибір мови програмування для реалізації інструменту було зроблено на користь Java. В контексті розробки такого засобу вона має ряд важливих переваг:

- Технологія Java Database Connectivity (JDBC), яка надає універсальний інтерфейс взаємодії з різноманітними СКБД. Для виконання запитів до

тієї чи іншої системи керування потрібно мати лише підходящий драйвер та вказати коректні дані для підключення. Завдяки розповсюдженості JDBC відповідні драйвери існують для більшості відомих СКБД. Вони підтримуються безпосередньо розробниками системи або спільнотою її користувачів. Отже, це полегшить додавання абсолютно різних СКБД до еталонного тесту.

- Широкі можливості з паралелізму та роботи з багатьма потоками. Гнучкі пули потоків, різноманітні методи їх контролю та синхронізації, наявність вбудованих колекцій та обгорток над стандартними типами, що призначені для роботи у паралельному середовищі – все це спрощує реалізацію еталонного тестування у декілька потоків.
- Віртуальна машина Java (JVM) реалізована на багатьох операційних системах. Таким чином, написаний на Java додаток може з легкістю бути запущений на різних системах, що робить його кросплатформеним. Оскільки СКБД можуть бути розгорнуті як на Windows, так і на Linux, можливість запуску еталонного тесту на обох системах є дуже важливою.

Збирання проекту, що має непросту структуру та ряд залежностей є доволі складним та кропітким завданням. У випадку з Java є ряд інструментів, що автоматизують та полегшують цей процес. Основними їх представниками є Apache Ant, Apache Maven та Gradle. Для цього проекту було обрано Maven. Використання системи збирання полегшує взаємодію з репозиторієм Maven Central, що містить більшість необхідних бібліотек. Їх підключення у проект не потребує ручного завантаження та додавання окремих JAR-файлів.

Для логування в додатку буде використано найпростішу реалізацію інтерфейсу SLF4J під назвою SLF4J Simple Provider. За замовчуванням вона виводить всі події до System.err. Цю поведінку, рівень детальності логів та їх формат можна змінити за допомогою файлу конфігурації. Для потреб застосунку таких можливостей буде цілком достатньо.

Ще однією залежністю фреймворку буде бібліотека, що призначена для зручного зчитування аргументів командного рядка, під назвою JCommander. За допомогою спеціальних анотацій вона дозволяє зіставляти аргументи відповідним полям класу. Бібліотека підтримує зчитування різноманітних типів даних та генерує зрозумілі для користувача помилки, в разі нестачі потрібних параметрів або надання їм некоректних значень.

2.4 Структура проекту

Розроблений застосунок використовує класичну монолітну архітектуру та основні принципи об’єктно-орієнтованого програмування.

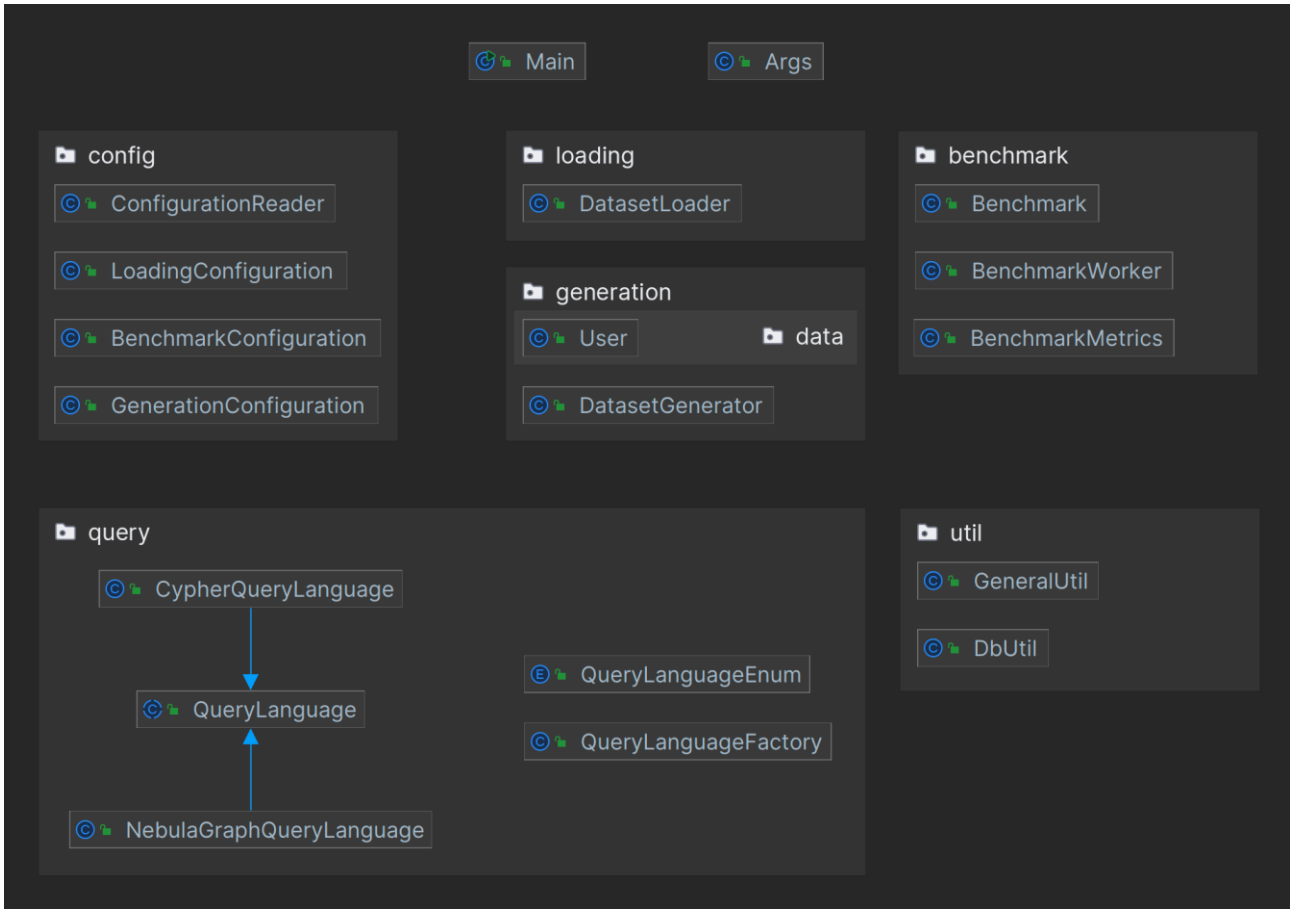

Рисунок 2.7 Діаграма класів проекту

На рисунку 2.7 зображено діаграму класів у проекті. Класи розділені по пакетах згідно зі своїм призначенням.

Клас Main є вхідною точкою застосунку та відповідає за запуск інших компонентів. В залежності від значення переданого параметру, Main може запустити три різні режими: генерація, завантаження та еталонне тестування.

```
@Parameter(names = {"-c", "--configuration"}, description = "Path to configuration file.", required = true)
private String configurationFile;
```

Рисунок 2.8 Анотація для встановлення зв’язку поля з параметром

Клас Args призначений для зберігання значень аргументів командного рядка, що були передані до програми. Він містить анотації згаданої раніше бібліотеки JCommander. Приклад такої анотації зображено на рисунку 2.8. Аргументи зчитуються засобами тієї ж бібліотеки у класі Main.

Класи DbUtil та GeneralUtil, що входять до пакету util, містять допоміжні методи, пов’язані з базою даних та загального призначення відповідно.

```
# Folder which have edges.txt and nodes.txt
dataset-folder = dataset

# After this many loaded entities display current progress
progress-interval = 100

# Language of queries sent to db
query-language = cql
```

Рисунок 2.9 Приклад файлу конфігурації

Пакет config складається з класів, що відповідають за зчитування та зберігання поточної конфігурації. ConfigurationReader безпосередньо зчитує та валідує налаштування кожного з режимів роботи програми. Конфігураційний файл являє собою пари виду ключ-значення. Фрагмент такого файлу наведено на рисунку 2.9. Розбір файлу забезпечується вбудованим класом Properties. Результатом виклику відповідних методів класу ConfigurationReader є екземпляри класів GenerationConfiguration, LoadingConfiguration, BenchmarkConfiguration. Кожен з яких зберігає налаштування, що потрібні для реалізованих режимів роботи.

```
package edu.ukma.query;

5 usages
public class QueryLanguageFactory {

    3 usages
    public QueryLanguage getLanguage(QueryLanguageEnum language) {
        return switch (language) {
            case CQL -> new CypherQueryLanguage();
            case NGQL -> new NebulaGraphQueryLanguage();
        };
    }

}
```

Рисунок 2.10 Клас QueryLanguageFactory

За виконання запитів до СКБД відповідають класи у пакеті query. З метою спрощення додавання до фреймворку підтримки баз даних, що використовують різні мови запитів, було створено абстрактний клас QueryLanguage з набором абстрактних методів, необхідних для завантаження даних та проведення тестування. Кожен нащадок класу імплементує ці методи, використовуючи коректний синтаксис для мови запитів, що додається. Аби спростити додавання та використання нових нащадків класу, використано патерн програмування під назвою Factory. Він передбачає спеціальний метод, якому делегується процес створення екземпляру. Однією зі складових патерну є перелічуваний тип даних (enum) QueryLanguageEnum, до якого входять ідентифікатори реалізованих у фреймворці мов запитів. Також створено клас QueryLanguageFactory, що складається з єдиного метода getLanguage, що приймає на вхід значення створеного enum, а повертає відповідний екземпляр класу-нащадку (рисунок 2.10).

Таким чином, щоб додати підтримку нової мови запитів, необхідно виконати такі кроки:

- створити клас-нащадок QueryLanguage
- надати реалізацію абстрактних методів, забезпечивши виконання ними коректних завдань

- додати ідентифікатор нової мови до QueryLanguageEnum
- доповнити оператор switch у методі getLanguage класу QueryLanguageFactory

В рамках роботи фреймворк було забезпечено підтримкою мов Cypher Query Language (CQL) та NebulaGraph Query Language (nGQL) для подальшого використання у процесі еталонного тестування. Відповідні класи з реалізацією запитів цими мовами, що входять до пакету query: CypherQueryLanguage, NebulaGraphQueryLanguage.

```java
package edu.ukma.generation.data;

8 usages
public class User implements Comparable<User> {
    2 usages
    private final int id;
    4 usages
    private final int friendsNumber;

    1 usage
    public User(int id, int friendsNumber) {
        this.id = id;
        this.friendsNumber = friendsNumber;
    }

    6 usages
    public int getId() {
        return id;
    }

    2 usages
    public int getFriendsNumber() {
        return friendsNumber;
    }

    @Override
    public int compareTo(User o) {
        return this.friendsNumber - o.friendsNumber;
    }
}
```

Рисунок 2.11 Клас User

За процес генерації набору даних для тестування відповідають класи в пакеті generation. Клас DatasetGenerator має метод, що генерує набір даних згідно з налаштуваннями, які надає користувач. Згенеровані дані зберігаються до текстових файлів у директорії, заданої користувачем. Клас User в субпакеті data репрезентує користувача соціальної мережі. Він містить поле з числовим ідентифікатором та кількістю друзів, згідно зі значенням якої відбувається генерація зв’язків (рисунок 2.11).

Пакет loading складається з єдиного класу DatasetLoader. Клас призначений для завантаження згенерованих даних до СКБД. Він зчитує дані з файлів заданої директорії та використовує для запитів екземпляр абстрактного класу QueryLanguage, конкретна реалізація якого отримується через його фабрику. Це дозволяє лишати метод завантаження незмінним при додаванні нових мов.

Врешті-решт, за процес тестування відповідає пакет benchmark. Він складається з трьох класів: Benchmark, BenchmarkWorker та BenchmarkMetrics. Перший з них є основним. На нього покладається процес ініціалізації тесту та запуску потоків, що відправлятимуть запити до СКБД. BenchmarkWorker являє собою клас окремого робочого потоку в тесті, тому реалізує інтерфейс Runnable. Він, як і DatasetLoader, використовуючи переваги патерну Factory, для запитів послуговується екземпляром абстрактного QueryLanguage, отримуючи ті ж переваги. Останнім класом є BenchmarkMetrics. Згідно з назвою, він відповідає за метрики еталонного тесту. Він складається з ряду колекцій та змінних, придатних для безпечної роботи з багатьма потоками. Спільний екземпляр цього класу передається всім потокам. За допомогою відповідних методів потоки оновлюють або передають різні показники, з яких наприкінці формується результат бенчмарку, який складається з метрик, що були описані у підрозділі 2.2.

Останнім, що варто згадати, розглядаючи структуру проекту, є файл pom.xml (від Project Object Model). Цей файл є частиною системи збирання Maven та містить об’єктну модель проекту: версію, назву, версію Java, список

залежностей тощо. В ньому перелічені бібліотеки, згадані в попередньому розділі, а також бібліотеки, що реалізують доступ до СКБД технологією JDBC, які ще називають драйверами. Більш того, в цьому файлі налаштовується процес збирання застосунку до архіву JAR для подальшого запуску.

Зібравши проект разом з усіма залежностями до JAR-файлу, застосунок стає можливим запустити на будь-якій машині зі встановленим оточенням Java.

2.5 Використання фреймворку

Використання фреймворку передбачає наявність зібраного архіву JAR, що запускатиме main-метод та міститиме всі необхідні залежності. Також у системі користувача має бути встановлене середовище виконання Java. Управління застосунком відбувається за допомогою аргументів командного рядку та змісту конфігураційного файлу.

```
java -jar <filename.jar> [options]
```

Рисунок 2.12 Синтаксис запуску програми з JAR

На рисунку 2.12 показано команду для запуску програми з JAR-файлу. Замість filename.jar користувач вказує назву свого файлу, а замість options потрібні аргументи, що будуть передані застосунку. Аргументи являють собою набір іменованих параметрів, що можуть бути позначені одинарним дефісом і назвою з однієї літери (-h) або подвійним дефісом та назвою, яка складається з декількох літер (--help). Вони можуть містити або не містити значення, вказаного користувачем (-o output.txt). Порядок параметрів неважливий.

```
benchmark-graph-db>java -jar benchmark-graph-db.jar -h
Usage: <main class> [options]
  Options:
    --clear
      Clear database before loading dataset. Used in loading mode
      Default: false
  * -c, --configuration
      Path to configuration file.
    -h, --help
      Print usage
    -m, --mode
      Application mode. Available options: generation, loading, benchmark
      Default: benchmark
```

Рисунок 2.13 Довідка застосунку

В застосунку є такі обов’язкові та опціональні параметри:

- Режим роботи. Позначається як -m або --mode. Приймає значення від користувача. Відповідає за те, у якому режимі буде запущено програму. Передбачає три коректних значення: generation, loading, benchmark – представляють режими генерації, завантаження даних та еталонного тестування відповідно. Є опціональним (у разі відсутності йому буде надано значення за замовчуванням benchmark).
- Конфігураційний файл. Позначається як -c або --configuration. Приймає значення від користувача. Відповідає за надання застосунку шляху до конфігураційного файлу (як відносного, так і абсолютного). Є обов’язковим параметром.
- Довідка. Позначається як -h або --help. Не приймає значень від користувача та є опціональним. Якщо вказано, застосунком буде виведено текст довідки, що містить список параметрів та їх опис (рисунок 2.13).
- Очистка СКБД. Позначається як --clear. Необов’язковий параметр, що не приймає значень від користувача. У разі його використання базу даних буде очищено перед завантаженням набору даних у режимі loading.

Кожен режим роботи передбачає характерний набір параметрів у конфігураційному файлі у форматі ключ-значення.

Параметр	Опис	Допустиме значення	Обов’язковий	Значення за замовчуванням
output-folder	Директорія, куди буде збережено згенерований набір даних	Рядок, що містить коректний шлях до директорії в відносному або абсолютному форматі	Так	Відсутнє
users-number	Кількість користувачів, що буде згенеровано	Ціле додатне число	Ні	1000
max-friends	Максимальна кількість друзів у одного користувача	Ціле додатне число, що менше за значення параметра users-number	Ні	20
min-friends	Мінімальна кількість друзів у одного користувача	Ціле додатне число, що менше або дорівнює значенню параметра max-friends	Ні	1

Таблиця 2.1 Параметри конфігурації режиму роботи generation

Запустивши програму у режимі generation (за допомогою відповідного аргументу командного рядка) та вказавши аналогічним способом файл конфігурації, що містить параметри, описані в таблиці 2.1, буде виконана генерація набору даних та його збереження до директорії, вказаної в output-folder у вигляді двох текстових файлів: nodes.txt та edges.txt.

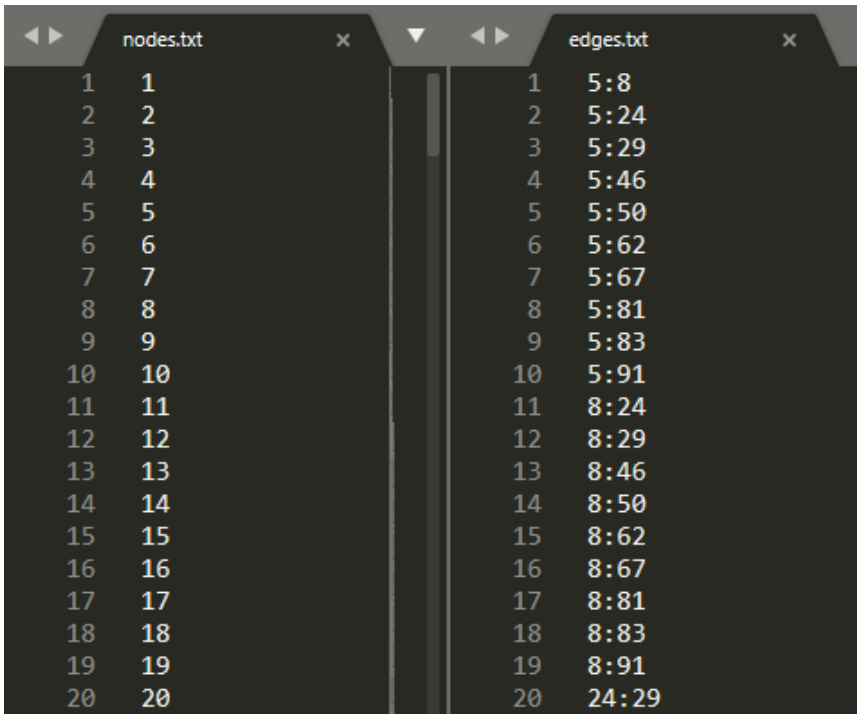

Рисунок 2.14 Фрагмент згенерованого набору даних

Текстовий файл nodes.txt складається з числових ідентифікаторів всіх користувачів, кількість яких залежить від users-number. Під час генерації кожен користувач отримує випадкову кількість друзів у межах від min-friends до max-friends. Дружба представляє собою односторонній направлений зв’язок від одного користувача до іншого та записується у файл edges.txt у форматі числових ідентифікаторів зв’язаних користувачів, що розділені двокрапкою. Фрагмент цих файлів наведено на рисунку 2.14.

Завдяки максимально простому формату згенерованих даних для подальшого застосування можна використати власний набір, що буде отриманий з реальної бази даних за умови його перетворення в потрібний вигляд. Також можна написати та використати власний алгоритм генерації. Єдине обмеження, про яке варто при цьому пам’ятати – всі числові ідентифікатори, що використовуються в ребрах (edges) мають бути присутніми у вузлах (nodes).

Параметр	Опис	Допустиме значення	Обов’язковий	Значення за замовчуванням
dataset-folder	Директорія, що містить набір даних для завантаження у СКБД	Рядок, що містить коректний шлях до директорії в відносному або абсолютному форматі	Так	Відсутнє
progress-interval	Після такої кількості завантажених об’єктів відображати поточний прогрес	Ціле додатне число	Ні	10000
query-language	Мова запитів, що буде використана для завантаження даних	Ідентифікатор мови запитів з наявною у проекті реалізацією. В рамках роботи: CQL, NGQL	Ні	CQL
db-url	URL цільової СКБД для підключення технологією JDBC	Коректний URL	Так	Відсутнє
db-auth	Визначає, чи потребує цільова СКБД авторизації за допомогою облікових даних	Значення типу boolean	Ні	false
db-user	Ім’я користувача для підключення до СКБД	Рядок	Ні	null
db-password	Пароль для підключення до СКБД	Рядок	Ні	null

Таблиця 2.2 Параметри конфігурації режиму роботи loading

```
benchmark-graph-db>java -jar benchmark-graph-db.jar -m loading -c loading.properties --clear
[2024-04-13 17:40:04:149] INFO org.neo4j.driver.internal.DriverFactory - Direct driver instance 111156771 created
for server address localhost:7687
[2024-04-13 17:40:04:498] INFO main - Cleared all data
[2024-04-13 17:40:04:499] INFO main - Loading nodes...
[2024-04-13 17:40:04:595] INFO main - Loaded 1000 nodes
[2024-04-13 17:40:04:599] INFO main - Nodes loaded in 96 ms
[2024-04-13 17:40:04:600] INFO main - Loading edges...
[2024-04-13 17:40:06:273] INFO main - Loaded 500 edges
[2024-04-13 17:40:07:391] INFO main - Loaded 1000 edges
[2024-04-13 17:40:08:735] INFO main - Loaded 1500 edges
[2024-04-13 17:40:09:706] INFO main - Loaded 2000 edges
[2024-04-13 17:40:10:713] INFO main - Loaded 2500 edges
[2024-04-13 17:40:11:232] INFO main - Edges loaded in 6631 ms
[2024-04-13 17:40:11:276] INFO main - Dataset was successfully loaded to database in 7400 ms
```

Рисунок 2.15 Приклад завантаження даних до СКБД Neo4j

Робота режиму loading полягає у заповненні цільової СКБД підготовленими даними. В цьому режимі фреймворк підключається до бази даних за допомогою JDBC та послідовно виконує запити на додавання вершин та ребер, зчитаних з файлів nodes.txt та edges.txt відповідно. Параметри, що можуть бути налаштовані користувачем, описано у таблиці 2.2. Окрім них, в режимі завантаження доступно використання згаданого раніше аргументу командного рядка --clear, що відповідає за попередню очистку СКБД.

Параметр	**Опис**	**Допустиме значення**	**Обов'язковий**	**Значення за замовчуванням**
threads	Кількість потоків, що паралельно відправлятимуть запити до СКБД	Ціле додатне число	Ні	1
requests	Кількість запитів, що буде відправлено до СКБД під час тестування	Ціле додатне число	Ні	1000
warmup-time	Проміжок часу перед початком тесту в секундах протягом якого потоки відправляють запити до СКБД, але не збирають показники	Ціле додатне число	Ні	0

users-number	Кількість користувачів у СКБД, що тестується. Еквівалентно максимальному ідентифікатору користувача.	Ціле додатне число	Так	Відсутнє
depth	Максимальна глибина пошуку друзів	Ціле додатне число	Ні	5
progress-interval	Після такої кількості виконаних запитів відображати поточний прогрес	Ціле додатне число	Ні	10000
query-language	Мова, що буде використана для відправки запитів до СКБД	Ідентифікатор мови запитів з наявною у проекті реалізацією. В рамках роботи: CQL, NGQL	Ні	CQL
db-url	URL цільової СКБД для підключення технологією JDBC	Коректний URL	Так	Відсутнє
db-auth	Визначає, чи потребує цільова СКБД авторизації за допомогою облікових даних	Значення типу boolean	Ні	false
db-user	Ім’я користувача для підключення до СКБД	Рядок	Ні	null
db-password	Пароль для підключення до СКБД	Рядок	Ні	null

Таблиця 2.3 Параметри конфігурації режиму роботи benchmark

Дочекавшись успішного завантаження набору даних, користувач може перейти до виконання еталонного тестування (режиму benchmark). Для фреймворку цей режим є найважливішим та має широкий список параметрів у своїй конфігурації. Їх описано у таблиці 2.3. Частина параметрів, що пов’язані з мовою запитів та даними підключення до СКБД є аналогічними режиму loading, проте є кілька, на яких варто зупинитися детальніше.

Перш за все, це параметр threads, що відповідає за кількість потоків тесту. Від цього значення сильно залежать отримані результати, адже при ефективній обробці паралельних запитів системою керування базами даних, що тестується, можуть зрости показники. Всі запити розподіляються між потоками, в залежності від швидкості їх виконання.

Одним з факторів, що впливає на час, за який буде проведено тест, є сумарна кількість запитів, адже тест закінчується лише тоді, коли вказаний об’єм запитів буде виконано. Вона вказується параметром requests. Більша кількість запитів зменшує можливу похибку результатів, нівелюючи вплив випадкових факторів.

Для збільшення достовірності результатів у бенчмарку передбачено час на розігрів кешу та інших можливих механізмів пришвидшення запитів. Він регулюється параметром warmup-time. Застосунок протягом вказаного часу перед тестом відправляє необмежену кількість запитів, не враховуючи час їх виконання до загального результату. Це дозволяє СКБД розігнатись до максимальної швидкості наявними механізмами та зменшує можливість отримання повільніших результатів на старті, що спотворило б реальні показники. Така практика широко застосовується у функціоналі тесту LinkBench [7].

Отже, для отримання об’єктивних результатів потрібно дотримуватись наступних порад:

- Базі даних потрібно дати достатньо часу на розігрів. Зробити це можна, вказавши warmup-time (кілька хвилин).

- Тестування потрібно проводити на великій кількості запитів, щоб мінімізувати вплив випадкових чинників та отримати стабільну швидкодію СКБД.
- Оберіть найкращу кількість потоків для вашої конфігурації, орієнтуючись на кількість ядер, логічних процесорів та провівши попередні тести з різними налаштуваннями.
- За можливості, зверніться до адміністратора СКБД для кращого налаштування системи на вашому обладнанні.

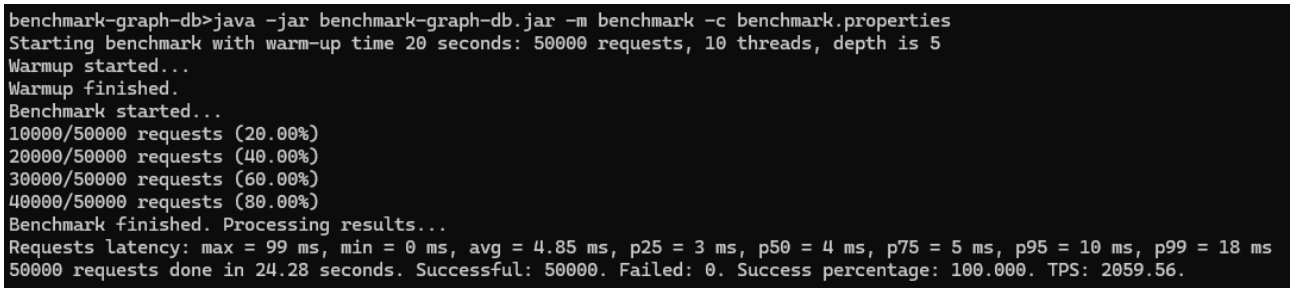

Рисунок 2.16 Ілюстративний приклад еталонного тестування СКБД Neo4j

На рисунку 2.16 наведено ілюстративний приклад проведеного еталонного тестування засобами фреймворку. З його допомогою можна побачити, у якому вигляді користувачу надаються метрики, що були описані у попередньому підрозділі.

Підсумовуючи, використання фреймворку передбачає наявність середовища Java та зібраного архіву JAR. Його налаштування здійснюється за допомогою аргументів командного рядку та конфігураційних файлів. Для отримання об’єктивних результатів потрібно дотримуватися вищевказаних рекомендацій.

Отже, використовуючи запропонований еталонний тест та фреймворк для тестування, що його реалізує, можна порівняти продуктивність СКБД, підтримка яких була забезпечена в ході розробки. Також за потреби архітектура застосунку дозволяє розширити список підтримуваних систем. Використання фреймворку є простим та не потребує особливих навичок.

РОЗДІЛ 3. ТЕСТУВАННЯ ГРАФОВИХ СКБД

З використанням розробленого фреймворку, проведемо еталонне тестування найпопулярніших графових СКБД та проаналізуємо отримані результати.

Rank			DBMS	Database Model	Score		
Apr 2024	Mar 2024	Apr 2023			Apr 2024	Mar 2024	Apr 2023
1.	1.	1.	Neo4j	Graph	44.47	+0.11	-7.13
2.	2.	2.	Microsoft Azure Cosmos DB	Multi-model	29.85	-0.54	-5.23
3.	3.	3.	Aerospike	Multi-model	6.10	-0.41	-0.30
4.	4.	4.	Virtuoso	Multi-model	4.20	-0.19	-2.04
5.	5.	5.	ArangoDB	Multi-model	3.77	-0.45	-1.03
6.	6.	6.	OrientDB	Multi-model	3.27	-0.11	-0.79
7.	↑ 8.	↑ 9.	GraphDB	Multi-model	3.10	+0.19	+0.76
8.	↓ 7.	↑ 11.	Memgraph	Graph	3.00	-0.09	+0.88
9.	9.	↓ 7.	Amazon Neptune	Multi-model	2.58	-0.24	-0.11
10.	10.	10.	NebulaGraph	Graph	2.12	-0.24	-0.05

Рисунок 3.1 Рейтинг графових СКБД [16]

Системи керування базами даних для тестування було обрано на основі рейтингу продуктів, що мають підтримку графової моделі даних як первинної, з веб-сайту DB-Engines. З нього було обрано три СКБД, що займають найвищі позиції та є суто графовими: Neo4j, Memgraph та NebulaGraph (рисунок 3.1, виділені жовтим) [16].

Еталонне тестування проводитиметься на пристрої під управлінням ОС Windows 10, що має оперативну пам’ять обсягом 16 гігабайтів та 4-ядерний процесор. Операційна система та всі СКБД встановлені на NVMe SSD-накопичувач. Для запуску систем керування, що не підтримують Windows нативно, було використано образи для Docker від розробників та технологію WSL 2.

СКБД використовуватимуться без додаткових налаштувань з параметрами за замовчуванням. Для обов’язкових налаштувань буде використано значення рекомендовані офіційною документацією. Застосовування методів оптимізації запитів буде описано у відповідних підрозділах, присвячених тестуванню окремих систем.

Буде використано набір даних, що згенеровано фреймворком. Він складається зі 100 тисяч користувачів (вершин), що мають від 40 до 60 друзів, які складають приблизно 2.5 мільйони ребер (кількість користувачів, помножена на середню кількість друзів та поділена на 2). Набір даних суттєво менший, ніж набір максимального розміру з експериментів у згаданій раніше книзі «Neo4j in Action», але достатньо великий, враховуючи можливості доступних обчислювальних потужностей [15].

Кожен тест запускатиметься з попереднім розігрівом СКБД тривалістю 5 хвилин та виконуватиме 1000 запитів. Це допоможе підвищити об’єктивність результатів.

Розроблений фреймворк надає можливість паралельної відправки запитів у декілька потоків. Для порівняння всі тести буде проведено у однопоточному та багатопоточному режимі. Під багатопоточним режимом мається на увазі тестування у 8 потоків. Така кількість показала найкращі результати на використовуваному обладнанні в результаті експериментальної перевірки.

Пошук друзів відбуватиметься на глибині 3, 4 та 5. Тестувати на глибині 2 не є доцільним, оскільки час виконання запитів буде занадто малим для надійного порівняння. Тести глибиною більше 5 не мають сенсу через переважну відсутність нових вузлів у результаті на використаному наборі даних.

Попередні експерименти показали повторюваність результатів, тому потреба у виконанні тестування декілька разів відсутня.

3.1 Neo4j

Першою СКБД на черзі еталонного тестування буде Neo4j.

Neo4j – графова система керування базами даних, перша версія якої була розроблена у 2002 році. Є однією з перших СКБД з графовою моделлю даних та однією з найпопулярніших. Написана на мові Java та наразі доступна у двох виданнях: Community – видання з відкритим вихідним кодом, що має ряд обмежень, та Enterprise – комерційна версія з ширшими можливостями [17].

Для проведення тестування було використано видання Enterprise, безкоштовна розробницька ліцензія до якого надається разом з інструментом управління Neo4j Desktop. СКБД має повну підтримку операційної системи Windows, що дозволяє використовувати її, не вдаючись до засобів віртуалізації.

Основним способом взаємодії з Neo4j є мова Cypher. Cypher Query Language (CQL) – декларативна графова мова запитів, що була розроблена у 2011 році інженерами Neo4j як аналог мови SQL для графових СКБД. У розробленому інструменті вона використовується для завантаження даних та запитів на їх отримання під час проведення тестування [18].

```
CREATE INDEX user_id_index FOR (n:User) ON (n.id)
```

Рисунок 3.2 Запит для створення індексу в Neo4j

За замовчуванням пошук вершини на графі за значенням його атрибуту у Neo4j потребує повного перебору, що веде до лінійної складності. Це не матиме значного впливу на малому обсязі вузлів, але серйозно впливатиме на продуктивність з ростом їх кількості. Для оптимізації такої операції у Neo4j передбачені індекси. Перед завантаженням даних та виконанням тестів було створено індекс для атрибуту id, що дозволить отримувати будь-яку вершину за логарифмічний час. Команду, що була використана для його створення, проілюстровано на рисунку 3.2.

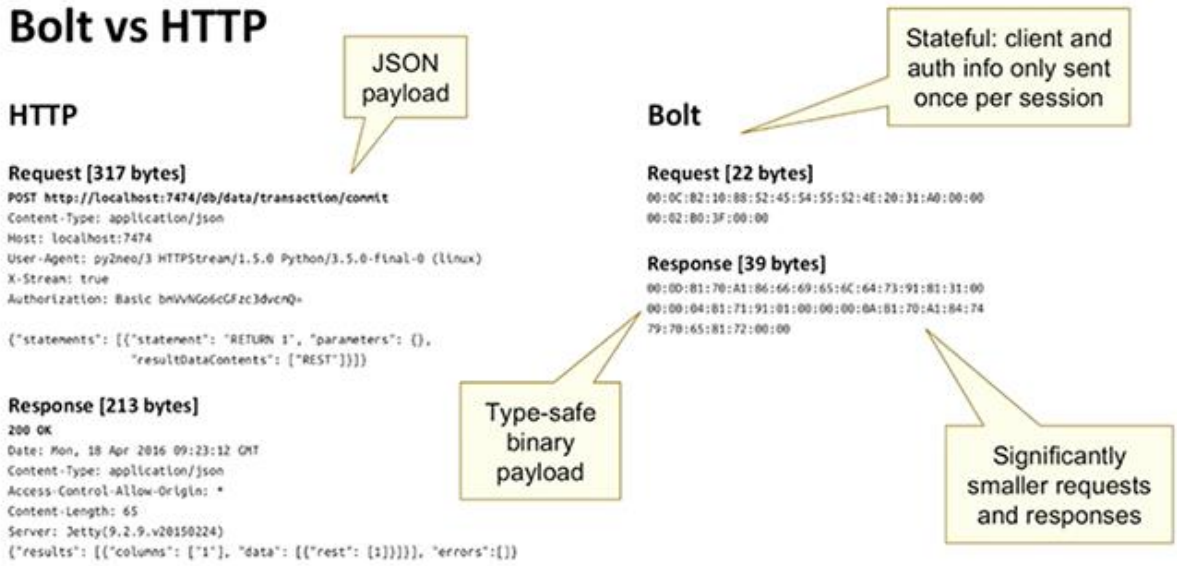

Рисунок 3.3 Порівняння протоколів Bolt та HTTP [19]

Для взаємодії з СКБД за допомогою технології JDBC було використано офіційний драйвер. Він підтримує комунікацію протоколами Bolt та HTTP. Bolt є низькорівневим протоколом, що забезпечує значно менший розмір запитів та відповідей, ніж HTTP, та потребує авторизації лише на початку сесії (рисунок 3.3). Оскільки розробниками рекомендується надавати перевагу протоколу Bolt, тестування проводилось з його використанням [19].

Глибина	**Час виконання (секунди)**	**TPS**	**Затримка окремих запитів (мілісекунди)**	
			95-й перцентиль	**Максимальна**
3	14.81	67.52	28	48
4	140.16	7.13	292	734
5	753.08	1.33	831	921

Таблиця 3.1 Результати еталонного тестування СКБД Neo4j в однопоточному режимі

Глибина	**Час виконання (секунди)**	**TPS**	**Затримка окремих запитів (мілісекунди)**	
			95-й перцентиль	**Максимальна**
3	4.51	221.88	68	114
4	44.20	22.62	775	1425
5	218.49	4.58	1973	2230

Таблиця 3.2 Результати еталонного тестування СКБД Neo4j в багатопоточному режимі

Отримані результати тестування наведено у таблицях 3.1 та 3.2. Можна побачити, що багатопоточний режим дає значний приріст у кількості транзакцій за секунду (приблизно у 3.3 рази більше на будь-якій глибині) та, відповідно, зменшує загальний час на виконання запитів. Це свідчить про більш ефективне використання ядер процесору та зменшення інтервалу між запитами. Варто відмітити, що при використанні багатьох потоків збільшується час обробки більшості транзакцій, що відображено у двох останніх стовпцях.

Очікувано, що зі збільшенням глибини пошуку час виконання запиту стає більшим, проте навіть на глибині 5, яка включає в себе майже всі вершини графу, більшість запитів виконується в межах однієї секунди для однопоточного режиму та двох секунд для багапоточного. Це видно зі значення 95-го перцентилю переведеного у секунди на відповідній глибини у таблицях 3.1 та 3.2.

Hardware requirement guidelines for personal use and software development

CPU	Intel x86-x64 Core i3 minimum, Core i7 recommended. AMD x86-x64, Mac ARM.
Memory	2GB minimum, 16GB or more recommended.
Storage	10GB SATA Minimum, SSD with SATA Express or NVMe recommended.

Hardware requirement guidelines for cloud environments

CPU	2vCPU minimum, 16+ recommended.
Memory	2GB minimum. Actual requirements depend on workloads. In some cases, it is recommended to use instances with memory that fits the size of the graph in use.
Storage	10GB minimum block storage, attached NVMe SSD recommended. Storage size depends on the size of the databases.

Hardware requirement guidelines for server-based, on-premise environments

CPU	Intel/AMD x86-x64. ARM64.
Memory	8GB minimum. Actual requirements depend on workloads. In some cases, it is recommended to use instances with memory that fits the size of the graph in use.
Storage	RAID/SAN or SSD with greater than 5000 IOPS. NVMe SSD is recommended. Storage size depends on the size of the databases.

Рисунок 3.4 Необхідне обладнання для роботи Neo4j в різних ситуаціях [34]

Для розуміння результатів тестування проведемо порівняння наявного обладнання з рекомендованим. Документація Neo4j надає інформацію для різних випадків використання: особистого (тобто на настільному комп’ютері), хмарного (оренди у стороннього провайдера) та серверного (на власних

потужностях). Ці вимоги до обладнання наведені на рисунку 3.4. Зважаючи на рекомендовані вимоги до особистого використання, обладнання, на якому проводилось тестування, їм цілком відповідає.

3.2 Memgraph

Memgraph – графова СКБД, що має відкритий вихідний код. Перша публічна версія була випущена у 2020 році [20]. Позиціонується як швидкий конкурент попередньо розглянутої системи Neo4j та надає сумісність з нею. У якості аргументів високої швидкості СКБД розробники називають використання мов C та C++, а також рушій сховища в оперативній пам’яті, забезпечуючи при цьому персистентність даних. Має відкрите Community та комерційне Enterprise видання [21].

Memgraph не має нативної підтримки операційної системи Windows, тому запуск системи здійснюється за допомогою офіційного образу для Docker. Всі тести виконувались на Community виданні Memgraph.

Як було зазначено раніше, Memgraph надає сумісність з Neo4j. Перш за все, це проявляється у використанні тієї ж мови запитів – Cypher. Memgraph орієнтується на специфікацію openCypher, що є відкритою реалізацією мови Cypher від розробників Neo4j, та прагне бути якнайближче до її типових реалізацій. Однак, є ряд функціоналу, що не підтримується у Memgraph, та інші відмінності, що потрібно враховувати у разі переходу з Neo4j [22].

```
CREATE INDEX ON :User(id)
```

Рисунок 3.5 Запит на створення індексу в Memgraph

Аналогічно з Neo4j для пришвидшення доступу до окремих вершин було створено індекс. Синтаксис його створення є однією з відмінностей у реалізації мов запитів. На рисунку 3.5 наведено команду для додавання відповідного індексу.

Ще однією складовою сумісності з Neo4j, що забезпечує Memgraph, є використання протоколу Bolt для комунікації з СКБД. За офіційною

документацією для підключення через JDBC використовується драйвер від системи Neo4j (власний не надається).

Глибина	**Час виконання (секунди)**	**TPS**	**Затримка окремих запитів (мілісекунди)**	
			95-й перцентиль	**Максимальна**
3	32.17	31.09	43	56
4	1312.21	0.76	1722	1908
5	61639.12	0.02	75344	77713

Таблиця 3.3 Результати еталонного тестування СКБД Memgraph в однопоточному режимі

Глибина	**Час виконання (секунди)**	**TPS**	**Затримка окремих запитів (мілісекунди)**	
			95-й перцентиль	**Максимальна**
3	10.39	96.23	109	152
4	418.55	2.39	4389	5039
5	23298.56	0.04	215784	216014

Таблиця 3.4 Результати еталонного тестування СКБД Memgraph в багатопоточному режимі

Пошук на глибині 3 демонструє доволі непогані показники швидкості та стабільні показники часу обробки окремих запитів. Збільшення глибини до 4-х сильно впливає на швидкість та погіршує показник транзакцій за секунду приблизно у 40 разів. Обробка запитів, що включають в себе пошук друзів глибиною 5 ще сильніше впливає на продуктивність. На ній час обробки більшості транзакції в однопоточному режимі сягає 75 секунд, що є серйозною деградацією.

Аналогічно кількість транзакцій за секунду при виконанні у декілька потоків на будь-якій глибині переважає однопоточний варіант у 2-3 рази, але збільшує середню тривалість запиту.

System requirements

Below are minimum and recommended system requirements for installing Memgraph.

	Minimum	Recommended
CPU	Server or desktop processor: Intel Xeon AMD Opteron/Epyc ARM machines or Apple M1 Amazon Graviton	Server processor: Intel Xeon AMD Opteron/Epyc ARM machines or Apple M1 Amazon Graviton
RAM	1 GB	≥ 16 GB ECC
Disk	1 GB	equally as RAM
Cores	1 vCPU	≥ 8 vCPUs (≥ 4 physical cores)
Network	100 Mbps	≥ 1 Gbps

Рисунок 3.6 Мінімальне та рекомендоване обладнання для роботи Memgraph [35]

Розглядаючи результати еталонного тестування, необхідно порівняти наявні потужності з рекомендованими. Рекомендовані наведені на рисунку 3.6. Вони передбачають наявність серверного процесору, що має 4 або більше фізичних ядра. Також рекомендується понад 16 гігабайтів оперативної пам’яті з підтримкою ECC. Проте для проведення тесту використовувався настільний процесор та пам’ять без підтримки ECC. До того ж Docker на ОС Windows може погіршити продуктивність у порівнянні з запуском на повноцінному дистрибутиві Linux. Це варто враховувати при розгляді результатів еталонного тестування.

3.3 NebulaGraph

NebulaGraph – СКБД з графовою моделлю даних, що була розроблена китайською компанією Vesoft. Перша версія системи була опублікована у 2019 році. Написана мовою C++ NebulaGraph характеризується розробниками як система для великих обсягів даних, що має розподілену архітектуру та

придатна до масштабування. Як і попередні продукти має два варіанти: відкритий Open Source та комерційний Enterprise [24].

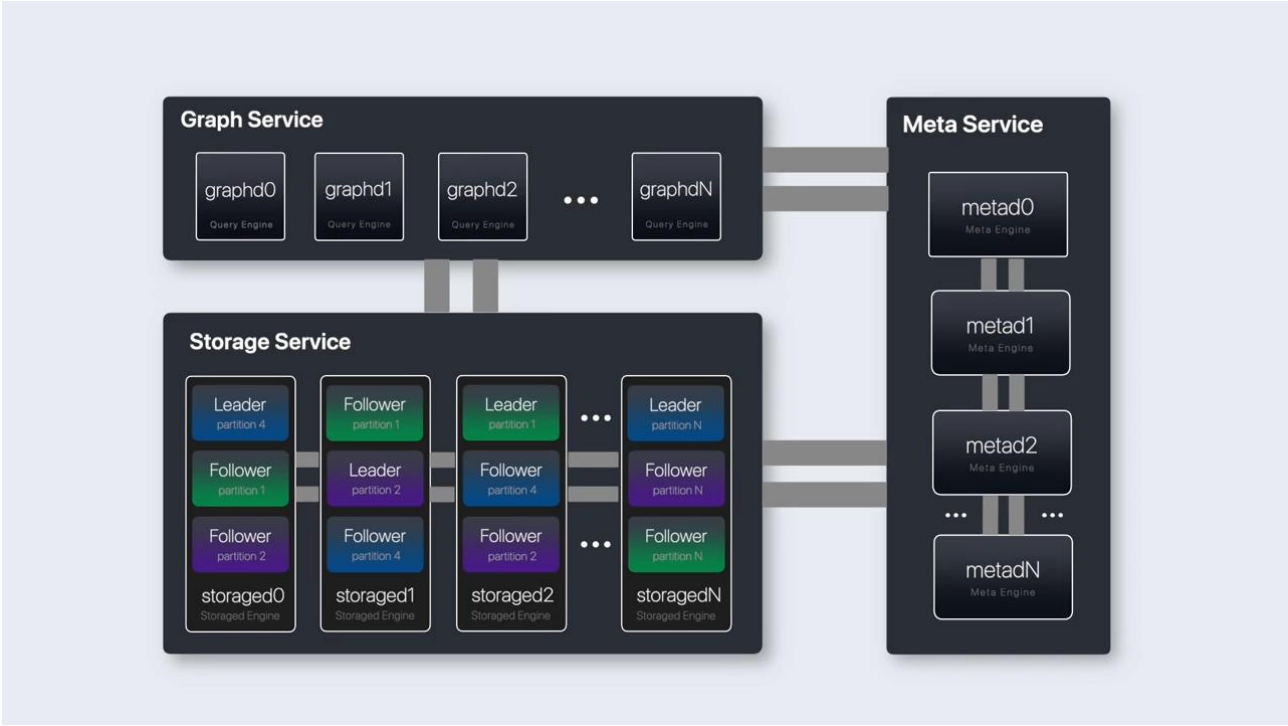

Рисунок 3.7 Архітектура NebulaGraph [24]

Для проведення тестування було використано видання Open Source. Нативна підтримка ОС Windows у NebulaGraph відсутня, тому для розгортання системи було використано офіційне розширення для Docker, що складається з набору образів, які формують архітектуру СКБД (рисунок 3.7).

NebulaGraph Query Language (nGQL) – декларативна мова запитів, що використовується для взаємодії з NebulaGraph. Вона надає часткову сумісність зі згаданим раніше проектом openCypher, але має ряд відмінностей та власних функцій, що виходять за його межі [25].

На відміну від попередніх СКБД NebulaGraph не потребує створення додаткового індексу для оптимізації пошуку вершини. Це зумовлено наявністю унікального ідентифікатору кожної вершини під назвою VID (Vertex ID), що задається користувачем під час додавання. Він може являти собою цілочисельне значення або рядок в залежності від налаштувань бази даних. Цей ідентифікатор індексується за замовчуванням, тому прямий доступ до вершини має високу ефективність на будь-якому об’ємі даних [26].

Для здійснення запитів до бази даних з розробленого фреймворку було використано офіційний драйвер для JDBC.

Глибина	**Час виконання (секунди)**	**TPS**	**Затримка окремих запитів (мілісекунди)**	
			95-й перцентиль	**Максимальна**
3	367.19	2.72	658	966
4	13768.35	0.07	18454	23371
5	-	-	-	-

Таблиця 3.5 Результати еталонного тестування СКБД NebulaGraph в однопоточному режимі

Глибина	**Час виконання (секунди)**	**TPS**	**Затримка окремих запитів (мілісекунди)**	
			95-й перцентиль	**Максимальна**
3	158.69	6.30	2154	3366
4	8306.33	0.15	26249	29837
5	-	-	-	-

Таблиця 3.6 Результати еталонного тестування СКБД NebulaGraph в багатопоточному режимі

В однопоточному тестуванні вже на глибині 3 час виконання запитів є доволі високим та наближається до однієї секунди. На глибині 4 кількість транзакцій за секунду падає майже у 40 разів, а час виконання окремих запитів займає понад 20 секунд. Як і у попередніх випадках, багатопоточний режим показує збільшення кількості транзакцій за секунду у 2-3 рази, збільшуючи затримку окремих операцій. Результати тестування на глибині 5 отримати не вдалось. При спробі виконання такого запиту СКБД повертала помилку та не змогла надати результат на доступному обладнанні.

Hardware requirements for production environments

Item	Requirement
CPU architecture	x86_64
Number of CPU core	48
Memory	256 GB
Disk	2 * 1.6 TB, NVMe SSD

Рисунок 3.8 Необхідне обладнання для роботи NebulaGraph у виробничому середовищі [36]

Варто зазначити, що розподілена архітектура призначена для використання на декількох машинах з великими обчислювальними потужностями. Зокрема, рекомендовані характеристики обладнання для використання NebulaGraph у виробничому середовищі наведено на рисунку 3.8. Вони в десятки разів переважають наявні у рамках цього тесту. До того ж СКБД розрахована на роботу на дистрибутивах Linux.

3.4 Аналіз результатів

Здійснивши еталонне тестування на різних системах керування базами даних, проведено порівняння результатів.

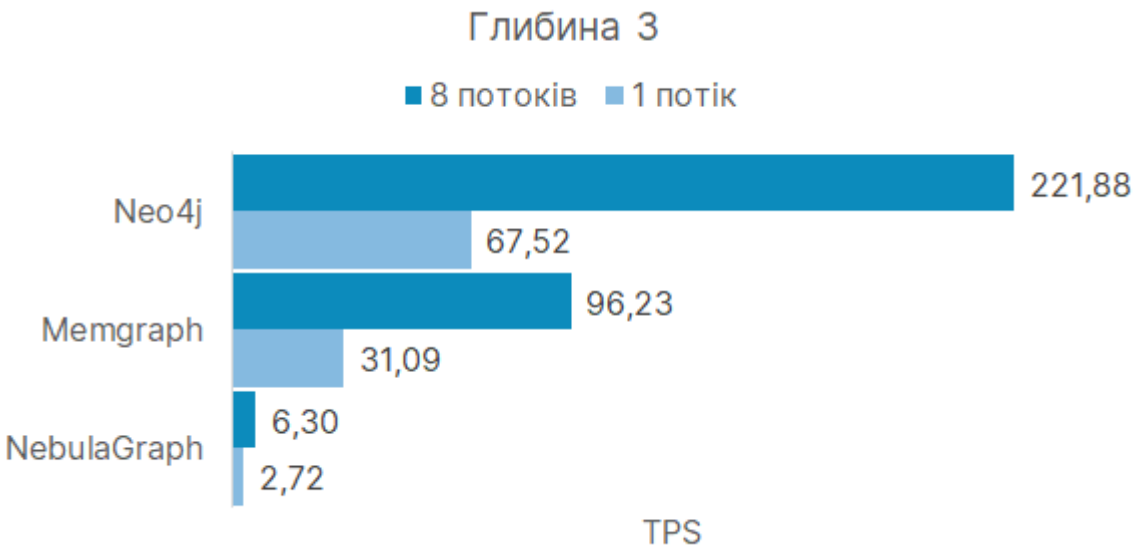

Рисунок 3.9 Порівняльний графік результатів тестування на глибині 3

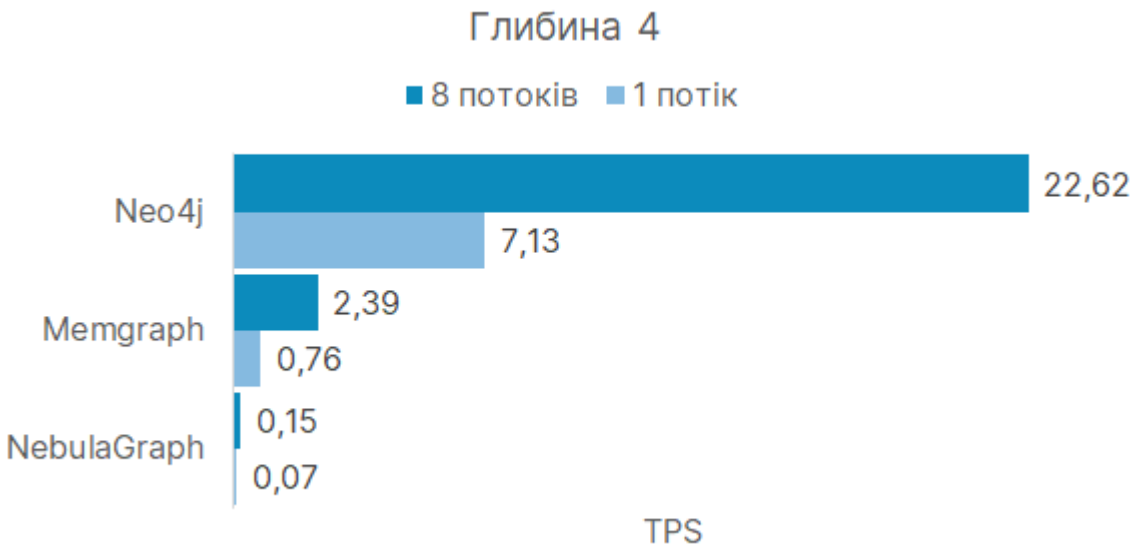

Рисунок 3.10 Порівняльний графік результатів тестування на глибині 4

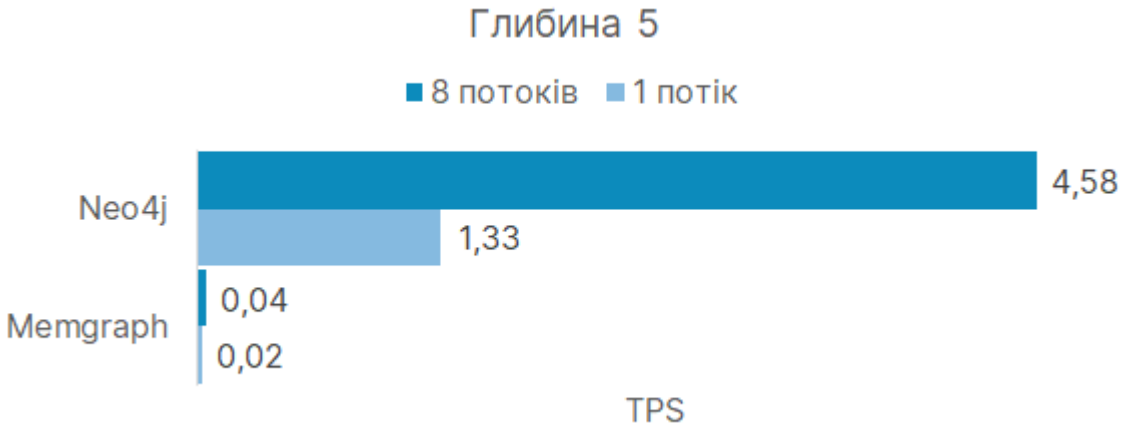

Рисунок 3.11 Порівняльний графік результатів тестування на глибині 5

На рисунках 3.9, 3.10 та 3.11 наведено діаграми, що порівнюють значення показника транзакцій за секунду (TPS) для кожної СКБД на різній глибині в режимі одного та багатьох потоків.

Загалом, з результатів очевидно, що безумовним лідером у всіх випадках є Neo4j та має серйозний відрив.

На глибині пошуку 3 Memgraph теж показує доволі непогані результати, залишаючись конкурентною проти показників Neo4j. NebulaGraph вже на цьому етапі показує дуже скромну продуктивність та обмежується кількома транзакціями за секунду.

За результатами тестування на глибині 4 Neo4j збільшила відрив від найближчого конкурента - системи Memgraph, яка показала серйозну деградацію продуктивності. Й без того невисокі показники NebulaGraph стали ще гіршими.

Глибина 5 унеможливила отримання результатів від NebulaGraph, як було зазначено раніше, тому порівняння проводиться лише між Neo4j та Memgraph. Бачимо, що TPS Neo4j все ще дозволяє їй обробляти понад 1 транзакцію за секунду. Зі сторони Memgraph сталось ще серйозніше погіршення продуктивності, тому для обробки кількох транзакцій їй знадобиться вже більше хвилини.

З наведених графіків також видно, що використання багатьох потоків на будь-якій з протестованих СКБД дозволяє обробляти більше транзакцій за одиницю часу й дає приріст цього показнику у декілька разів.

Інших доступних інструментів, що дозволяють одночасно порівняти між собою всі системи, що тестувалися, наразі немає. Більшість інструментів надає підтримку для Neo4j та інших продуктів, що не розглядалися в межах роботи, бо не є СКБД виду LPG або є не такими популярними.

Отже, за допомогою експериментального тестування було визначено, що найкращі показники на обладнанні невеликої потужності демонструє СКБД Neo4j. Проведено аналіз та відзначено деталі, що варто враховувати при розгляді отриманих показників кожної з систем. Окремо виділено позитивний вплив використання декількох потоків на загальну продуктивність всіх СКБД, що тестувалися.

ВИСНОВКИ ПО РОБОТІ

В роботі було оглянуто сферу еталонного тестування СКБД. Проаналізовано та порівняно ряд еталонних тестів, інструментів тестування. Окремо приділено увагу поточному стану еталонного тестування графових СКБД.

Застосувавши досвід існуючих рішень, було запропоновано власний еталонний тест, що призначений для порівняння продуктивності графових СКБД. Для його реалізації та проведення було розроблено фреймворк, що надає можливість генерувати набір тестових даних довільного розміру, автоматизовано завантажувати його до СКБД та проводити еталонне тестування, в результаті якого користувач отримує набір різних показників, що допомагають оцінити продуктивність СКБД та порівняти результати. Розроблений інструмент є легким у налаштуванні та використанні. Архітектура фреймворку передбачає можливість додавання підтримки нових СКБД у кілька кроків, що дозволяє розширювати спектр його застосування.

Ще одним результатом роботи є експериментальне тестування кількох популярних систем: Neo4j, Memgraph та NebulaGraph. Завдяки розробленому інструменту було проведено еталонне тестування згаданих СКБД та проаналізовано отримані результати.

Робота сприяє розвитку еталонного тестування графових баз даних. Втім, ця сфера залишає широкий простір для подальших досліджень.

СПИСОК ЛІТЕРАТУРИ

1. TPC History of TPC. TPC-Homepage. URL: https://www.tpc.org/information/about/history5.asp (date of access: 19.04.2024).

2. TPC-C Overview. TPC-Homepage. URL: https://www.tpc.org/tpcc/detail5.asp (date of access: 19.04.2024).

3. YCSB. ScyllaDB. URL: https://www.scylladb.com/glossary/ycsb/ (date of access: 19.04.2024).

4. Core Workloads. GitHub. URL: https://github.com/brianfrankcooper/YCSB/wiki/Core-Workloads (date of access: 19.04.2024).

5. Implementing New Workloads. GitHub. URL: https://github.com/brianfrankcooper/YCSB/wiki/Implementing-New-Workloads (date of access: 19.04.2024).

6. Running a Workload. GitHub. URL: https://github.com/brianfrankcooper/YCSB/wiki/Running-a-Workload (date of access: 19.04.2024).

7. LinkBench / T. G. Armstrong et al. the 2013 international conference, New York, New York, USA, 22–27 June 2013. New York, New York, USA, 2013. URL: https://doi.org/10.1145/2463676.2465296 (date of access: 19.04.2024).

8. OLTP-Bench / D. E. Difallah et al. Proceedings of the VLDB Endowment. 2013. Vol. 7, no. 4. P. 277–288. URL: https://doi.org/10.14778/2732240.2732246 (date of access: 19.04.2024).

9. MS SQL/Oracle Database Benchmark & Performance Testing Tool. Quest | IT Management | Mitigate Risk | Accelerate Results. URL: https://www.quest.com/products/benchmark-factory/ (date of access: 19.04.2024).

10. GitHub - akopytov/sysbench: Scriptable database and system performance benchmark. GitHub. URL: https://github.com/akopytov/sysbench/ (date of access: 19.04.2024).

11. osdldbt. GitHub. URL: https://github.com/osdldbt/ (date of access: 19.04.2024).

12. HammerDB Documentation. HammerDB. URL: https://www.hammerdb.com/docs/ (date of access: 19.04.2024).

13. pgbench. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/current/pgbench.html (date of access: 19.04.2024).

14. GitHub - Percona-Lab/tpce-mysql. GitHub. URL: https://github.com/Percona-Lab/tpce-mysql (date of access: 19.04.2024).

15. Neo4j in Action / A. Vukotic et al. Manning Publications, 2014. 304 p.

16. DB-Engines Ranking. DB-Engines. URL: https://db-engines.com/en/ranking/graph+dbms (date of access: 19.04.2024).

17. Company. Graph Database & Analytics. URL: https://neo4j.com/company/ (date of access: 19.04.2024).

18. Overview - Cypher Manual. Neo4j Graph Data Platform. URL: https://neo4j.com/docs/cypher-manual/current/introduction/cypher_overview/ (date of access: 19.04.2024).

19. Small N., Plantikow S. A Deeper Dive into Neo4j 3.0 Language Drivers - Graph Database & Analytics. Graph Database & Analytics. URL: https://neo4j.com/blog/neo4j-3-0-language-drivers/ (date of access: 19.04.2024).

20. About us. Memgraph. URL: https://memgraph.com/about-us (date of access: 19.04.2024).

21. Memgraph database. Memgraph. URL: https://memgraph.com/memgraphdb (date of access: 19.04.2024).

22. Differences in Cypher implementation. Memgraph. URL: https://memgraph.com/docs/querying/differences-in-cypher-implementations (date of access: 19.04.2024).

23. About vesoft Inc., the Creator of the Open-Source Distributed Graph Database. vesoft Inc. | Unleash the power of Interconnected Data with Graph Databases. URL: https://vesoft.com/about/ (date of access: 19.04.2024).

24. NebulaGraph Graph Database. Open Source Distributed Graph Database | NebulaGraph. URL: https://www.nebula-graph.io/ (date of access: 19.04.2024).

25. Overview - NebulaGraph Database Manual. NebulaGraph Docs. URL: https://docs.nebula-graph.io/3.6.0/3.ngql-guide/1.nGQL-overview/1.overview/ (date of access: 19.04.2024).

26. VID - NebulaGraph Database Manual. NebulaGraph Docs. URL: https://docs.nebula-graph.io/3.6.0/1.introduction/3.vid/ (date of access: 19.04.2024).

27. [2022 Nov] The LDBC Social Network Benchmark. Linked Data Benchmark Council. URL: https://ldbcouncil.org/docs/presentations/ldbc-snb-2022-11.pdf (date of access: 19.04.2024)

28. FinBench: The new LDBC benchmark targeting financial scenario. Linked Data Benchmark Council. URL: https://ldbcouncil.org/benchmarks/finbench/finbench-talk-16th-tuc.pdf (date of access: 19.04.2024)

29. GitHub - socialsensor/graphdb-benchmarks: Performance benchmark between popular graph databases. GitHub. URL: https://github.com/socialsensor/graphdb-benchmarks (date of access: 19.04.2024).

30. Lissandrini M., Brugnara M., Velegrakis Y. Beyond macrobenchmarks. Proceedings of the VLDB Endowment. 2018. Vol. 12, no. 4. P. 390–403. URL: https://doi.org/10.14778/3297753.3297759 (date of access: 19.04.2024).

31. Introduction to Benchgraph and its Architecture. Memgraph. URL: https://memgraph.com/blog/introduction-to-benchgraph-and-its-architecture (date of access: 19.04.2024).

32. TPC Current Specs. TPC-Homepage. URL: https://www.tpc.org/tpc_documents_current_versions/current_specifications5.asp (date of access: 19.04.2024).

33. GitHub - MillenniumDB/WDBench: Benchmark resources. GitHub. URL: https://github.com/MillenniumDB/WDBench (date of access: 19.04.2024).

34. System requirements - Operations Manual. Neo4j Graph Data Platform. URL: https://neo4j.com/docs/operations-manual/current/installation/requirements/ (date of access: 19.04.2024).

35. Install Memgraph. Memgraph. URL: https://memgraph.com/docs/getting-started/install-memgraph (date of access: 19.04.2024).

36. Resource preparations - NebulaGraph Database Manual. Redirecting. URL: https://docs.nebula-graph.io/3.6.0/4.deployment-and-installation/1.resource-preparations/ (date of access: 19.04.2024).