DOI 10.34185/1562-9945-4-153-2024-13
УДК 004.4

І.В. Андрюханов, Т.О. Коротєєва

МОДЕЛЬ СЕРВІСУ ГЕТЕРОГЕННОЇ РОЗПОДІЛЕНОЇ БАЗИ ДАНИХ ДЛЯ МАСШТАБУВАННЯ ПРОГРАМНИХ СИСТЕМ

Анотація. У статті розглянуто актуальність горизонтального масштабування програмних систем, які потребують використання СУБД для опрацювання великих об'ємів структурованих, JSON, геопросторових даних, при цьому, маючи вимоги забезпечення ACID властивостей та цілісності даних. Проведено аналіз існуючих продуктів, наведено їх основні переваги та недоліки, сформульовано проблему єдиної точки відмови, технічних обмежень, відсутності підтримки необхідного функціоналу. У якості її вирішення розроблено модель сервісу гетерогенної розподіленої бази даних, складено опис її характеристик та архітектури. З застосуванням технології Kubernetes проведено дослідження метрик продуктивності кластера БД на базі розробленої моделі, показники яких у підсумку мають кращі значення у порівнянні з розглянутими рішеннями.
Ключові слова: розподілені СУБД, кластер баз даних, OLTP, ACID, JSON, геопросторові дані, горизонтальне масштабування, продуктивність баз даних.

Постановка проблеми. Розвиток інформаційних технологій, їх масове використання для автоматизації процесів у фінансових, технологічних, медичних, наукових, державних та інших сферах сучасного світу, глобальна цифровізація послуг потребує створення складних програмних систем для опрацювання high-load навантажень на великих об'ємах даних, які можуть бути представлені в структурованих та неструктурованих форматах. Серед останніх варто відзначити суттєве збільшення JSON та геопросторових даних останніми роками. В свою чергу, цей факт ставить нові вимоги щодо потреб масштабування СУБД, яке може бути реалізовано за рахунок використання NoSQL рішень з підтримкою даного функціоналу. Однак, NoSQL бази даних підтримують лише неструктуровані формати, а їх архітектура збудована на основі BASE транзакційної моделі (відсутність миттєвої узгодженості), тому БД цього типу не можуть в повній мірі замінити реляційні для проектів, орієнтованих на OLTP та ACID властивості. Тому розробники вимушені використовувати декілька типів СУБД з різною кількістю екземплярів кожного, в

залежності від навантаження. Проте в цьому випадку лишаються наступні проблеми: відсутність масштабування однієї з реляційних баз даних [1, 2], а також диспетчеризації з'єднань на рівні сервісів бізнес логіки [1, 3], що у свою чергу ускладнює архітектуру серверних застосунків, їх масштабування, підтримку та супровід програмних систем.

Аналіз останніх досліджень і публікацій. Проаналізувавши відповідні тематичні статті та наукові джерела, було знайдено декілька рішень, які дозволяють розгортати розподілені кластери баз даних з підтримкою необхідного SQL функціоналу. Але, вони все одно мають певні недоліки: один вузол координатор, як вузьке місце всієї системи, відсутність глобального індексу – Citus DB, Greenplum DB [4, 5]. Cockroach DB не підтримує транзакцій високої складності, збережені процедури та тригери [6], просунуті можливості стосовно гео зон та бекапів доступні лише для enterprise версії. Volt DB має обмеження на розмір бази в залежності від розміру пам'яті, не підтримує зовнішні ключі, check constraints, JSON тип даних [7], постачається за платною ліцензією. В табл. 1 наведено відмінності між цими продуктами, їх основні переваги та недоліки.

Таблиця 1

Основні характеристики розподілених реляційних баз даних

Властивості	Citus DB	Greenplum DB	Cockroach DB	Volt DB
Workloads	OLTP	OLAP	OLTP	OLTP
Цінова модель	+	+	-	-
Розподіл даних	distributed value, size	evenly, randomly	ranges, geo (enterprise)	partition per node
Глобальний індекс	-	-	+	-
Резервування, гнучкість масштабування	-	-	+ (enterprise)	memory restriction
FK, check constraints	+	+	+	-
SQL proc., triggers	+	+	-	+
JSON data & operators	+	+	restricted operators	-
Геопросторові дані	+	+	+	+

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Мета дослідження. Метою дослідження є розробка моделі сервісу розподіленої системи управління базами даних, яка має підтримувати структуровані, JSON, геопросторові дані, ACID набір властивостей, надавати можливість прозорої інтеграції з клієнтом через єдиний API, при цьому забезпечуючи гнучке масштабування всіх компонентів програмної системи.

Програмна реалізація моделі. Враховуючи основні аспекти проектування розподілених баз даних, було розроблено модель сервісу гетерогенної розподіленої СУБД, основними вимогами реалізації якої є:

- З боку кластера – єдиний інтерфейс та різні його реалізації для проведення досліджень з різними БД продуктами;
- З боку клієнта – єдиний інтерфейс для прозорої взаємодії з інструментом тестування, однакові тестові сценарії для різних БД продуктів;
- Розділення всього функціоналу на окремі незалежні компоненти, взаємодія яких відбувається через внутрішні інтерфейси;
- Паралельна обробка запитів у багатопоточному режимі, можливість використання декількох екземплярів сервісів для гнучкого масштабування програмної системи;
- Збір та агрегація внутрішньої статистики здійснюється у фоновому режимі з мінімальним впливом на продуктивність кластера;
- Окрема БД для збереження метаданих, забезпечення її високого рівня надійності, доступності та продуктивності, низької вартості передачі даних від цієї БД до всіх точок використання в кластері.

На рис. 1 зображено архітектурну схему моделі, розробка якої виконана з застосуванням Massively Parallel Processing (MPP) підходу.

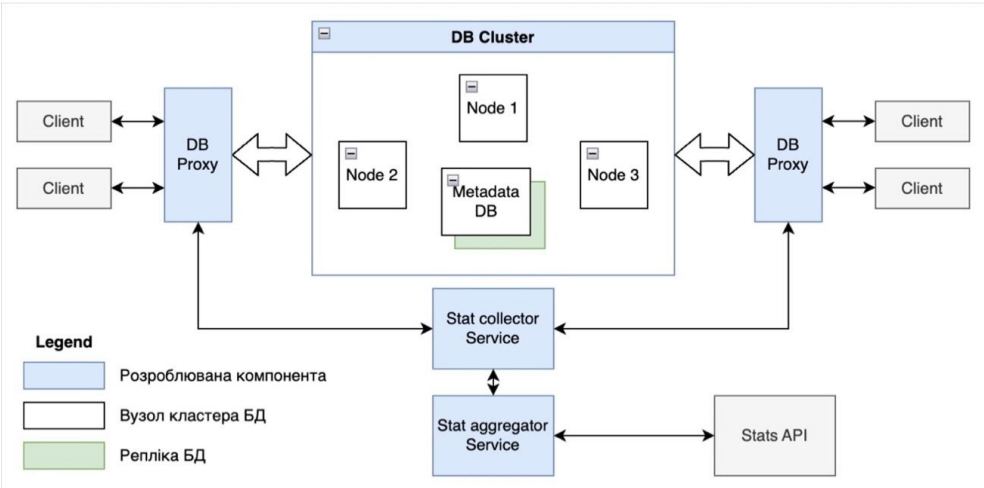

Рисунок 1 – Архітектурна схема моделі сервісу розподіленої системи керування базами даних

У якості опрацювання даних на кожному вузлі використовується PostgreSQL v15.3 з PostGIS розширенням - реляційна СУБД з підтримкою JSONB та Geometry типів даних. Для вузла з метаданими використовується масштабоване NoSQL рішення Redis Stack, яке розгортається в двох екземплярах: основна БД та додаткова репліка. Це вирішує проблему одного вузла координатора, в production експлуатації метадані можуть бути масштабовані на додаткові вузли кластера, при цьому забезпечується одночасне опрацювання запитів для декількох DB Proxy екземплярів. Розподіл даних між вузлами реалізовано на основі Hash-based sharding алгоритму. Він може бути застосований для одного або декілька полів distributed об'єкта, який уособлює сутність одного запису певного дата сету. Програмна система розроблена на основі Java 17 з використанням Spring фреймворку версії 6.

Характеристики програмного продукту:

- Забезпечення розподілу даних серед вузлів кластера прозоро для клієнта;
- Підтримка числових та текстових типів даних для distributed об'єкта;
- Підтримка JSON та геопросторових типів даних для атрибутів таблиць;
- Можливість паралельної обробки одного запиту на декількох вузлах кластера, агрегація результатів в одну відповідь для клієнта;
- Збір та агрегація статистики за наступними параметрами: номер вузла кластеру, БД сутність, група запитів. Метрики: кількість запитів, кількість задіяних унікальних записів БД сутності, середній час обробки з розрахунку розподілення на кожний запис;
- Генерація змістовних дата сетів заданого об'єму для кожної БД сутності;
- Створення та запуск сценаріїв тестування продуктивності для кожної БД сутності і декількох одночасних з'єднань, з можливістю здійснювати послідовно різні типи запитів та аналізувати їх успішність виконання, формувати статистику по групам запитів для кожного сценарію.

Зовнішні програмні інтерфейси:

- RESTful Client API – CRUD операції для інтеграції з інструментом тестування продуктивності з підтримкою JSON формату;
- RESTful API з підтримкою TSV формату для взаємодії з сервісом статистики.

Дослідження продуктивності розробленої моделі. Для створення кластеру БД з трьох вузлів та розгортання розробленої програмної системи було

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

застосовано Minikube – це легковагова реалізація Kubernetes для локального використання. Характеристики комп'ютера, на якому було проведено дослідження: MacBook Pro 2,9 GHz Intel Core i7, 16 GB RAM, 500 GB SSD, macOS Ventura 13.6. Створена віртуальна машина на базі гіпервізора Hyperkit: 4 vCPU, 6GB RAM, 20GB HDD. Для програмування сценаріїв навантаження та тестування продуктивності було використано інструмент Gatling, для генерації змістовних даних – бібліотеку Datafaker. Список дата сетів з їх характеристиками вказано у табл. 2. Для третього дата сету використано перелік географічних назв з сайту Science Base-Catalog з їх координатами у форматі загальноприйнятої системи просторової прив'язки (SRID) 4326, який представляє просторові дані з використанням координат довготи та широти на поверхні Землі, як визначено в стандарті WGS84. Кількість записів у 45000 в кожному дата сеті підібрано експериментально для цієї конфігурації, щоб запобігти використанню swap на віртуальній машині БД кластеру.

Таблиця 2

Опис характеристик дата сетів

Дата сет	Тип distributed value	Змінюване значення UPD сценарію	Тип полів для сценаріїв пошуку та UPD ключа
TextDS	Varchar(130)	Varchar(1300)	Int(8), Varchar(130)
JsonDS	Int(8)	JSON with 20 random properties	JSON key:val pair Varchar (6-12:10-20)
GeoDS	Int(8)	Geometry(POINT)	Int(8), Geometry(POINT)

Дослідження продуктивності роботи програмної системи засновано на відпрацюванні всіх сценаріїв для кожного дата сету протягом 120 секунд (окрім create та delete) для десяти одночасних БД з'єднань, при цьому дані з дата сету використовуються у випадковій послідовності. Сценарій create виконувався до повного відпрацювання дата сету, сценарій delete застосовувався для 5000 записів з кожного дата сету. Після відпрацювання кожного сценарію через прикладний програмний інтерфейс статистики (API) отримувались значення метрик продуктивності для подальшого аналізу. Зміст сценаріїв та їх послідовність запитів до бази даних вказано у табл. 3.

Таблиця 3

Перелік сценаріїв тестування продуктивності роботи кластера

Назва сценарію	Послідовність запитів до БД
Create	create
Update	updateByDVal, findMany, findById
Find	findByDVal, findMany, findById
FindJson	findByJson, findById
FindByPoint	findByPoint, findById
FindByDistance	findByDistance, findById
Delete	delete

На рис. 2 представлено порівняння результатів тестування власної розробленої моделі, Citus DB v12.1 with PostGIS v16, Cockroach DB v23.1 з застосуванням create сценарію.

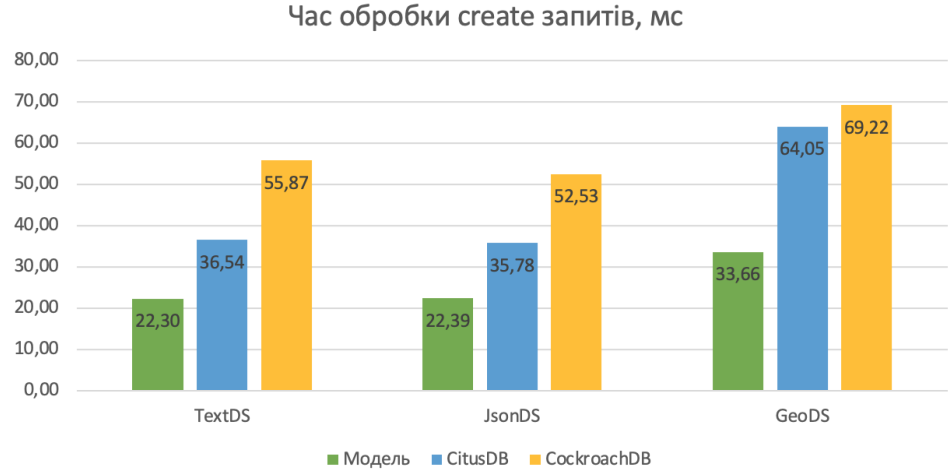

Рисунок 2 – Час обробки create запитів для всіх дата сетів

Розроблена модель має кращій показник продуктивності у 2 рази за обидва БД продукти для гео даних. Для інших типів даних – у 1,6 рази краще від Citus DB та у 2,35-2,5 від Cockroach DB. Для всіх БД продуктів створення запису з гео даними потребує у 1,24-1,8 більше часу ніж для інших типів даних.

Результати оновлення за distributed значенням представлено на рис. 3 у вигляді метрики часу обробки updateByDVal запитів з Update сценарію для всіх дата сетів.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

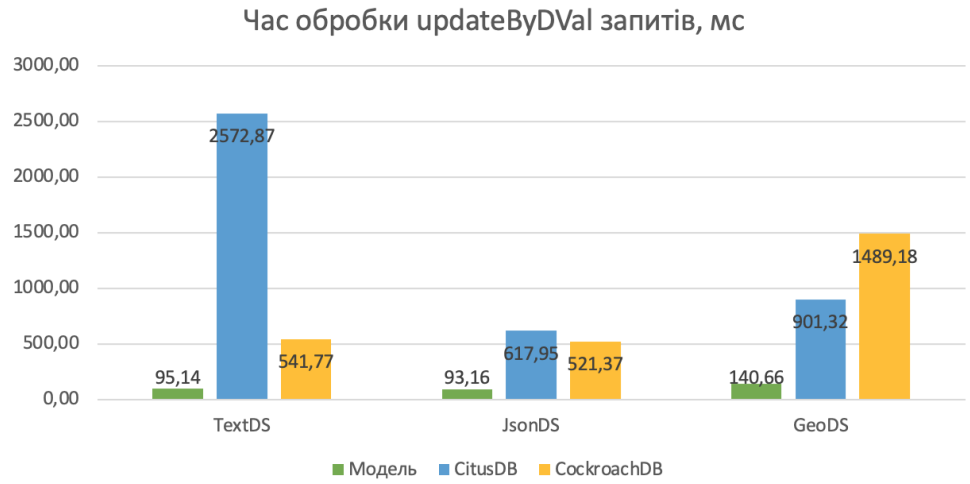

Рисунок 3 – Час обробки updateByDVal запитів для всіх дата сетів

Розроблена модель має кращі значення метрики продуктивності у порівнянні з іншими БД продуктами в 5,6-10,6 раз без урахування результату на TextDS дата сеті для Citus DB. Значення цієї метрики слід відокремити від інших, воно відрізняється у порівнянні з Cockroach DB у 4,7 рази та у 27 раз від значення відповідної метрики розробленої моделі. Незважаючи на створений індекс для distributed текстового поля TextDS дата сету, Citus DB має найгірший показник продуктивності запиту оновлення даних таблиць, записи яких розподіляються в кластері на основі іншого унікального індексу на текстовому полі, ніж на ідентифікаторі у вигляді integer primary key. Для дата сету, в якому у якості distributed значення використовується integer primary key, Citus DB показує набагато кращий результат, але все одно програє іншим БД продуктам. У порівнянні з попереднім тестом, зберігається тенденція того, що оновлення записів з гео даними потребує більше часу.

Дослідження метрики продуктивності для запитів, які повертають багато рядків наведено на рис. 4 у вигляді часу обробки findMany запиту з результатом у 100 рядків з того ж Update сценарію.

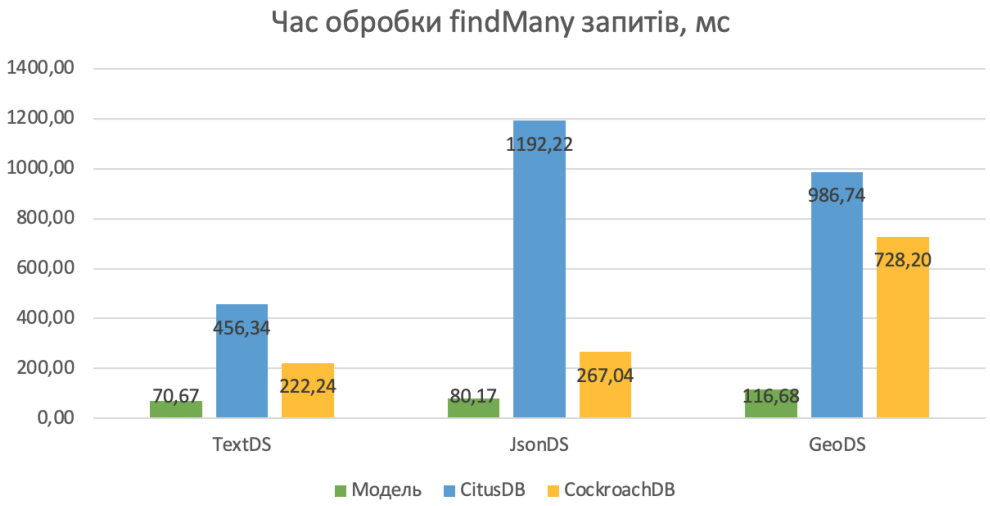

Рисунок 4 – Час обробки findMany запитів для всіх дата сетів

Не враховуючи метрику Citus DB для JsonDS дата сету, розроблена модель знову має кращий результат у 3,2-8,5 рази, а з врахуванням цієї метрики у 14,9 рази. Citus DB показує найгірший результат отримання JSON записів.

Метрики запитів для пошуку за атрибутами та ідентифікатором JSON даних, а також їх видаленню показано на рис. 5. Не враховуючи результати пошуку даних у Citus DB кластері, які знову мають найгірші значення, розроблена модель має найкращі показники у 3,4-4,7 рази від інших БД продуктів. Порядок розташування метрик за їх значеннями для delete запитів має ту ж послідовність, як і для create операцій, відповідно ці значення відрізняються між собою у 1,6-3,4 рази.

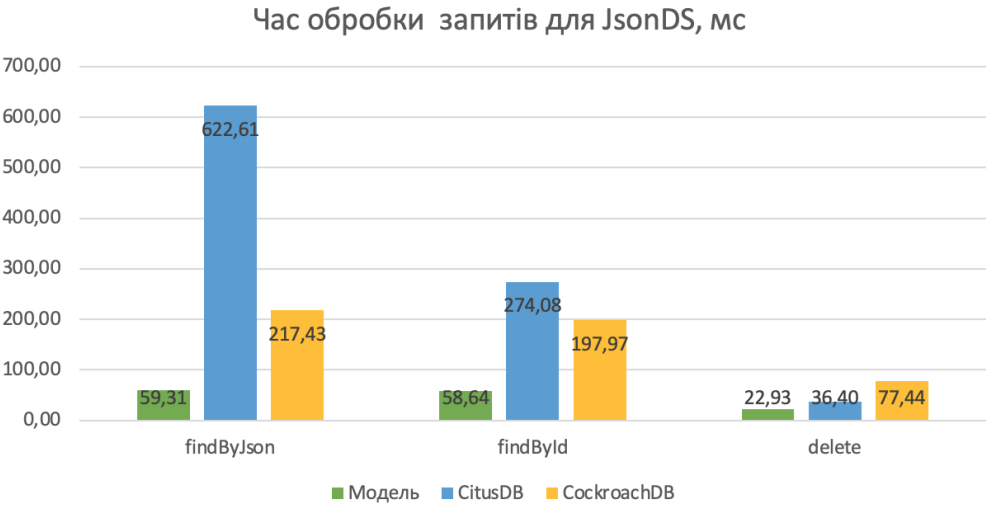

Рисунок 5 – Час обробки запитів для JsonDS дата сету

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Результати останнього тесту продуктивності запитів пошуку гео даних для різних кластерів показані на рис. 6.

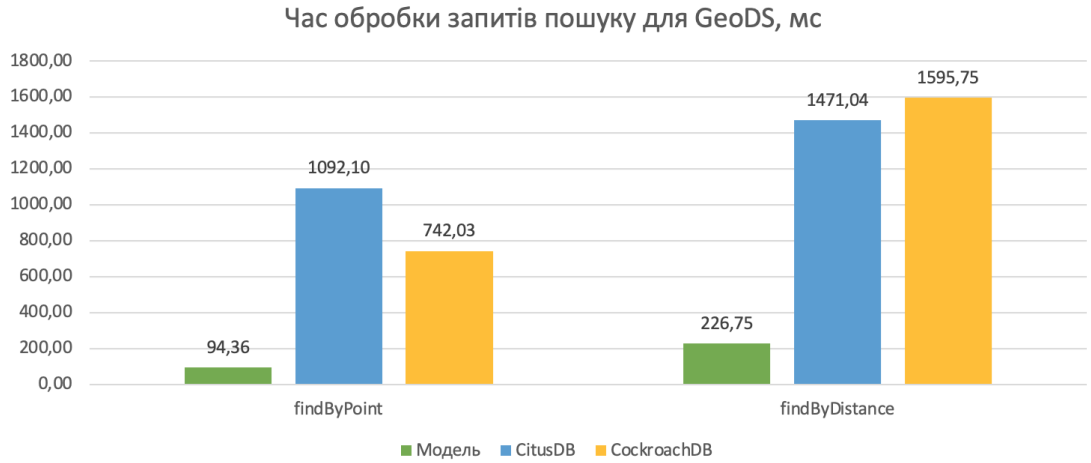

Рисунок 6 – Час обробки запитів пошуку для GeoDS дата сету

У цьому випадку метрики розробленої моделі у 7,9-11,6 разів краще від інших кластерів для запиту пошуку по координатах, у 6,5-7 разів краще для запиту пошуку за відстанню з сортуванням.

У табл. 4 наведено показники використання пам'яті на рівні віртуальної машини як мінімальні та максимальні значення по всім сценаріям тестування, а також сумарний розмір файлового сховища для даних кластера.

Таблиця 4

Використання апаратних ресурсів на рівні віртуальної машини

Кластер	Used memory		Free memory		Buff/cache		File store
	min	max	min	max	min	max	
Модель	1.0Gi	1.1Gi	1638Mi	1740Mi	2.9Gi	3.0Gi	1192Mi
Citus DB v12.1	1.0Gi	1.1Gi	822Mi	887Mi	3.9Gi	4.0Gi	697Mi
CockroachDB v23.1	2.7Gi	2.9Gi	142Mi	372Mi	2.6Gi	3.0Gi	1622Mi

Ці дані показують, що Cockroach DB потребує найбільше апаратних ресурсів, Citus DB має найменший розмір сховища, розроблена модель має кращі показники за об'ємом вільної пам'яті.

Висновки та перспективи подальших досліджень. У статті актуалізовано проблему масштабування СУБД для гео розподілених систем, які мають підтримувати структуровані та неструктуровані дані (JSON, geospatial) для проектів, орієнтованих на OLTP та ACID властивості. Проведено аналіз

існуючих програмних продуктів (Citus DB, Greenplum DB, Cockroach DB, Volt DB), наведено їх основні характеристики, переваги та недоліки.

Розроблена власна модель сервісу гетерогенної розподіленої БД, яка усуває проблему єдиної точки відмови, має єдиний інтерфейс для прозорої інтеграції з клієнтом, надає можливість гнучкого масштабування всіх компонентів програмної системи. Проведення дослідження продуктивності кластерів на основі аналогів Citus DB, Cockroach DB показує кращі результати для розробленої моделі. У подальшій роботі планується реалізувати перерозподіл даних на основі метрик продуктивності [8], а також провести дослідження з урахуванням фактору реплікації.

ЛІТЕРАТУРА / REFERENCES

1. Design and Development of a Relational Database Management System (RDBMS) with Open Source Tools for the Processing of Data Monitored in a Set of Photovoltaic (PV) Plants / D. Trillo-Montero et al. Applied Sciences. 2023. Vol. 13, no. 3. P. 1357. URL: https://doi.org/10.3390/app13031357 (date of access: 12.04.2024).

2. A framework and databases for measuring entrepreneurial ecosystems / E.Johnson, I. Hemmatian, L. Lanahan, A. M. Joshi. // Research Policy. – 2022. – no. 2.

3. Simanjuntak E., Surantha N. Multiple time series database on microservice architecture for IoT-based sleep monitoring system. Journal of Big Data. 2022. Vol. 9, no. 1. URL: https://doi.org/10.1186/s40537-022-00658-4 (date of access: 09.04.2024).

4. Wu A., Guo J., Yang P. Research on Data Sharing Architecture for Ecological Monitoring Using Iot Streaming Data. IEEE Access. 2020. Vol. 8. P. 195385–195397. URL: https://doi.org/10.1109/access.2020.3034466 (date of access: 12.04.2024).

5. Citus: distributed PostgreSQL for data-intensive applications / [U. Cubukcu, O. Erdogan, S. Pathak and other.]. // SIGMOD. – 2021. – P. 2490–2502.

6. Cockroach Labs Documentation Team. CockroachDB Docs. CockroachDB Docs. URL: https://www.cockroachlabs.com/docs/ (date of access: 12.04.2024).

7. Pina E., Sá F., Bernardino J. NewSQL Databases Assessment: CockroachDB, MariaDB Xpand, and VoltDB. Future Internet. 2022. Vol. 15, no. 1. P. 10. URL: https://doi.org/10.3390/fi15010010 (date of access: 12.04.2024).

8. Fedorchuk Y., Andriukhanov I. The method of redistribution of data in a cluster environment. Youth, education and science through today's challenges : XII International Science Conference, Bordeaux France, 4 Dec. 2023. P. 450–453. URL: https://eu-conf.com/wp-content/uploads/2023/11/YOUTH-EDUCATION-AND-SCIENCE-THROUGH-TODAYS-CHALLENGES.pdf

Received 15.05.2024.

ISSN 1562-9945 (Print)
ISSN 2707-7977 (Online)

Accepted 16.05.2024.

The service model of a heterogeneous distributed database for software system scaling

The article explores the topicality of horizontal scaling for software systems that require the use of database management systems (DBMS) to process large volumes of structured, JSON, and geospatial data, while maintaining requirements for ACID properties and data integrity. An analysis of existing products is conducted, outlining their main advantages, disadvantages and identifying issues such as single points of failure, technical limitations, and lack of support for necessary functionality. To solve these issues, a service model of a heterogeneous distributed database is developed, providing a description of its characteristics and architecture. Using Kubernetes technology, performance metrics of the database cluster based on the developed model are investigated, demonstrating superior performance compared to the products discussed.

Keywords: distributed DBMS, database cluster, OLTP, ACID, JSON, geospatial data, horizontal scaling, database performance.

Андрюханов Ігор Вікторович – магістр, випускник кафедри програмного забезпечення, Інституту комп'ютерних наук та інформаційних технологій, Національний університет «Львівська політехніка».

Коротєєва Тетяна Олександрівна – кандидат технічних наук, доцент кафедри програмного забезпечення, Інституту комп'ютерних наук та інформаційних технологій, Національний університет «Львівська політехніка».

Andriukhanov Ihor – Master's graduate from the Department of Software, Institute of Computer Science and Information Technologies, Lviv Polytechnic National University.

Korotyeyeva Tetyana – Ph.D. in Technical Sciences, Associate Professor at the Department of Software, Institute of Computer Science and Information Technologies, Lviv Polytechnic National University.