УДК 004.65

А.В. Миргородський[1]
О.В. Романюк[1]

АНАЛІЗ АРХІТЕКТУРНИХ ПРИНЦИПІВ СУЧАСНИХ РОЗПОДІЛЕНИХ СИСТЕМ КЕРУВАННЯ БАЗАМИ ДАНИХ

[1]Вінницький національний технічний університет

Анотація

Проаналізовано високорівневі архітектурні принципи сучасних розподілених СКБД, визначено відмінності між гомогенною та гетерогенною архітектурами розподілених БД; досліджено механізми фрагментації та реплікації даних в БД, проаналізовано їх роль в сучасних розподілених СКБД.

Ключові слова: СКБД, розподілена БД, архітектура розподіленої БД, гомогенна та гетерогенна архітектура, фрагментація даних, реплікація даних.

Abstract

The paper provides brief information about the high-level architectural principles of modern distributed DBMSs, considers the differences between homogeneous and heterogeneous distributed database architectures, investigates the mechanisms of data fragmentation and replication in the database, and analyzes their role in modern distributed DBMSs.

Keywords: DBMS, distributed database, distributed database architecture, homogeneous and heterogeneous architecture, data fragmentation, data replication.

Вступ

Обчислювальні системи, що складаються з декількох логічно або фізично відокремлених вузлів, не є чимось новим для сфери інформаційних технологій. Такий підхід, в тій чи іншій формі, застосовується ще з минулого століття для створення кластерних або розподілених систем. Проте, варто відмітити, що технологічно та архітектурно розподілені системи минулого та сьогодення значно відрізняються. Розвиток програмного та апаратного забезпечення, а саме удосконалення технологій віртуалізації та контейнеризації або ж поява дешевшої та потужнішої інфраструктури в раніше недоступних місцях, спричинив популяризацію хмарних та кордонних обчислень (англ. cloud computing та edge computing відповідно) [1].

Аналогічні зміни відбуваються і в сфері розподілених баз даних. Архітектурні підходи поступово еволюціонують та адаптуються до сучасних сценаріїв застосування. Тому, питання дослідження архітектурних принципів проектування сучасних розподілених систем керування базами даних є актуальним і може допомогти виявити подальші напрямки розвитку цього класу інформаційних систем.

Метою дослідження є вивчення архітектурних принципів систем керування розподіленими базами даних.

Об'єктом дослідження є процес проектування систем керування розподіленими базами даних.

Предмет дослідження – архітектура систем керування розподіленими базами даних.

Основна частина

Розподілена база даних – це база даних (БД), що складається з декількох фізично або логічно (у випадку застосування віртуалізації або інших технік мережевої ізоляції) відокремлених вузлів. При цьому взаємодія з БД є прозорою для кінцевого користувача й виглядає як використання традиційної централізованої системи керування БД [2]. Тобто, клієнт БД не знає про розподілену природу системи та/або про стан окремих її вузлів.

Розподілені бази даних можуть архітектурно відрізнятись як за загальним дизайном системи керування, так і за окремими підходами в зберіганні та обробці даних. Більш традиційною класифікацією з точки зору високорівневої архітектури є розподіл на однорідні та різнорідні системи. Однорідні (також гомогенні, англ. homogenous) системи передбачають застосування однакового ПЗ та

моделі даних на всіх вузлах. Різнорідні (також гетерогенні, англ. heterogeneous) системи можуть мати різне програмне забезпечення на вузлах (наприклад, змінені компоненти для підтримки певних апаратних можливостей) або ж різні моделі даних, щоб різні вузли могли по-різному обробляти дані. В більш широкому розумінні гетерогенна розподілена база даних може бути додатковою системою контролю, що об'єднує декілька незалежних БД й встановлює додаткові правила комунікації між цими системами [2]. Схематичне відображення цих архітектур наведено на рисунку 1.

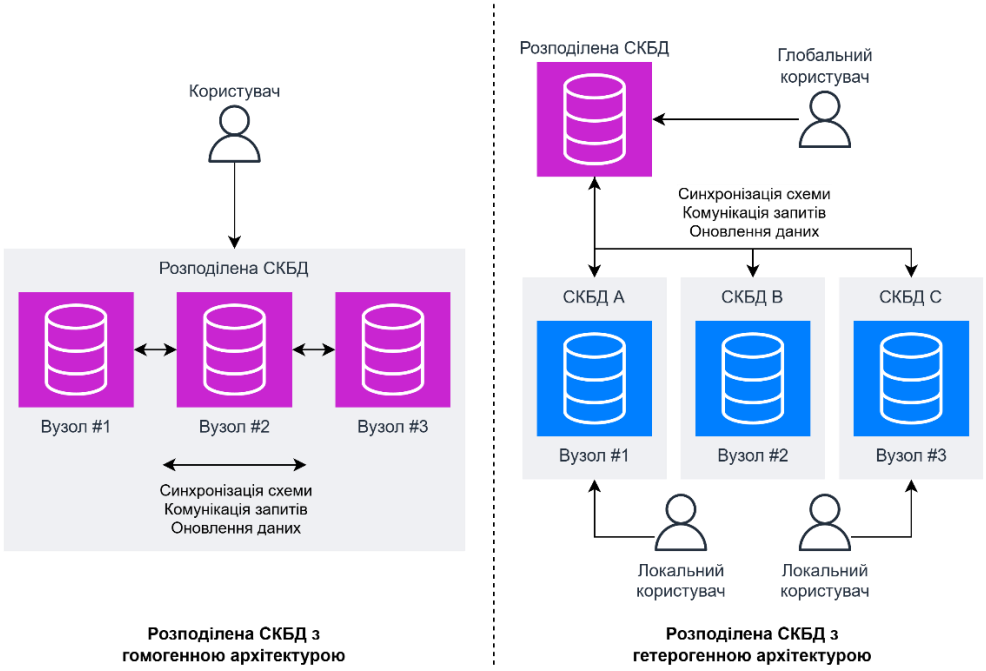


Рисунок 1 – Гомогенна та гетерогенна архітектури

Крім того, однорідні розподілені бази даних іноді підрозділяють на автономні та неавтономні в залежності від поведінки вузлів в системі. Сервери автономної системи керування розподіленими базами даних обробляють вхідні запити незалежно одне від одного, а сервери неавтономної системи мають окремий центральний вузол або ж інший механізм узгодження оновлення даних в РБД (розподіленій базі даних) в цілому. Прикладом може бути архітектура на основі алгоритму консенсусу, що наведена на рисунку 2.

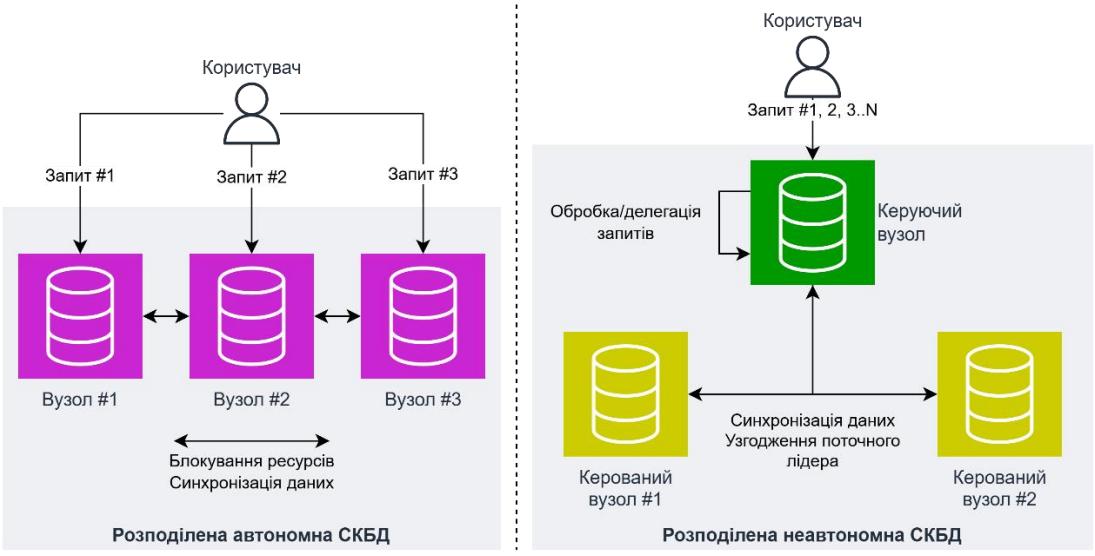


Рисунок 2 – Приклади архітектури однорідної автономної та неавтономної розподіленої БД

На даний момент ця класифікація досі присутня, проте більша частина популярних розподілених баз даних є гомогенними за своєю архітектурою та передбачають однакову або хоча б схожу поведінку всіх вузлів системи.

Важливими елементами високорівневої архітектури розподіленої БД є також механізми зберігання й розподілу даних, до яких можна віднести фрагментацію та реплікацію даних. Ці механізми також стали ще більш ефективними з популяризацією хмарних і кордонних обчислень, коли доступ до більш потужного апаратного та мережевого забезпечення став більш простим [2].

Фрагментація даних (англ. data sharding) дозволяє розподілити великі обсяги даних між кількома вузлами системи. Архітектура із застосуванням фрагментації може досягти більшої продуктивності, масштабованості та відмовостійкості. Завдяки фрагментації забезпечується зменшення обсягу даних, які потрібно обробити кожному вузлу, тому це також є засобом оптимізації доступу до них.

Фрагментація передбачає поділ даних на логічні частини (фрагменти), які можуть зберігатися на різних вузлах [3]. Фрагментація не повинна впливати на роботу застосунків, а кожен фрагмент має бути самодостатнім, що дозволяє виконувати запити до нього без доступу до інших фрагментів. Крім того, кожен елемент даних має належати лише до одного фрагмента (виняток – більш рідкісні випадки застосування фрагментації разом з реплікацією, коли окремі фрагменти все-таки можуть дублюватися на декількох вузлах).

Фрагментація може бути реалізована кількома способами залежно від структури даних всередині кожного фрагмента. Два основні типи фрагментації – це горизонтальна та вертикальна.

При горизонтальній фрагментації кожен фрагмент містить підмножину рядків таблиці, що задовольняють певну умову [4]. Наприклад, фрагмент може містити записи всіх користувачів з певного регіону або тих, хто має однакові/схожі налаштування профілю. Це робить запити, що стосуються тільки цієї підмножини рядків, дуже швидкими й простими, проте збільшує час на обробку глобальних запитів, що вимагають інформацію з усієї таблиці. Горизонтальна фрагментація є більш поширеною серед класичних реляційних баз даних.

При вертикальній фрагментації кожен фрагмент містить підмножину стовпців таблиці, але кожен елемент зберігає свій унікальний ідентифікатор для визначення відповідного рядка-власника [4]. Вертикальна фрагментація та інші подібні механізми часто застосовуються в NoSQL (нереляційних) БД, таких як Amazon DynamoDB – окремі групи атрибутів можуть зберігатися на одному вузлі для більш швидкого доступу. Такий тип зберігання даних також часто використовується в OLAP (англ. online analytical processing) системах, де перевага надається швидкій вибірці даних по окремих атрибутах/стовпцях для виконання аналітичних запитів або формування автоматичних звітів, а доступ до повних даних кожного рядка є менш використовуваним. Приклад роботи СКБД з обома типами фрагментації наведено на рисунку 3.

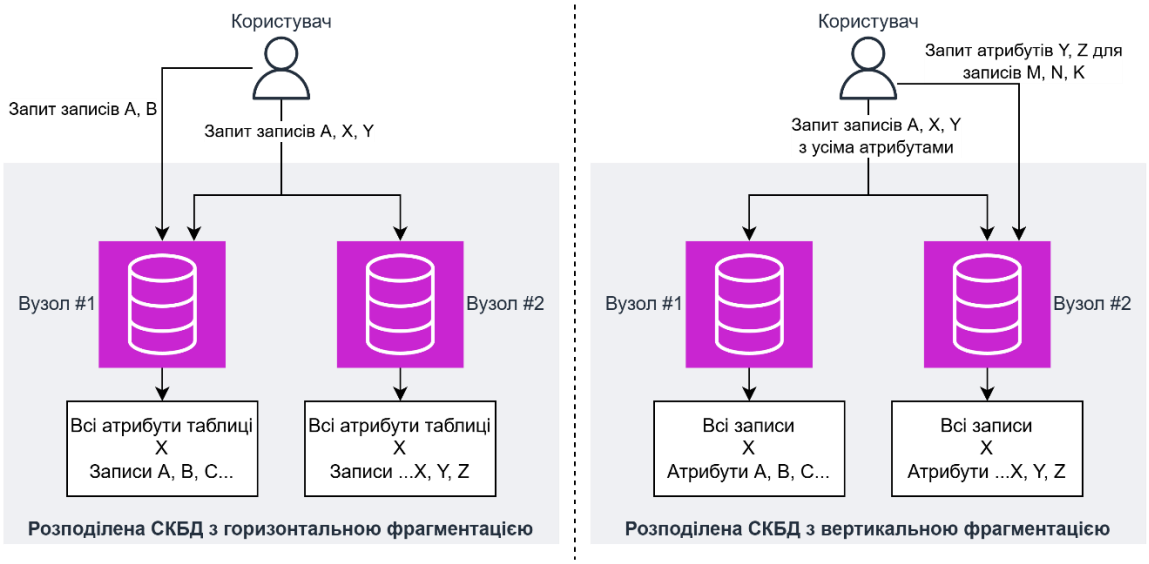


Рисунок 3 – Архітектура розподіленої СКБД з горизонтальною та вертикальною фрагментацією

Розподілені бази даних, що застосовують механізми фрагментації даних в своїй архітектурі, також можуть мати й різні динамічні показники ефективності роботи. Можна виділити два основних алгоритми розподілу фрагментів – статичний та динамічний. При статичному розподілі розміщення даних визначається ще на етапі налаштування РБД або ж в момент першого отримання даних нового типу чи створенні нового вузла, після чого існуючі фрагменти завжди залишаються на своїх вузлах. Простота такого підходу відбирає можливість подальшої адаптації системи – ефективність РБД може значно знизитись при нерівномірному розподілі фрагментів між вузлами. Динамічний розподіл вимагає реалізації додаткової системи моніторингу та балансування фрагментів – операційна вартість використання такої системи потенційно вище, але вона зможе адаптуватися до змін в структурі даних БД та високого навантаження до окремих фрагментів.

Таким чином, фрагментація дозволяє оптимізувати доступ до даних в розподіленій БД – застосування різних видів поділу даних та розміщення фрагментів дозволяє налаштувати систему для максимальної ефективності під конкретні задачі та сценарії використання. Тому, фрагментація даних може стати важливим елементом архітектури при проектуванні вузькоспеціалізованої розподіленої системи керування базами даних.

Проте, основний недолік фрагментації даних – це загальна вразливість системи до виходу з ладу окремих вузлів, що може призвести до втрати доступу до окремих рядків або стовпців/атрибутів. Високорівнева архітектура бази даних також повинна враховувати відмовостійкість – це одна з найважливіших характеристик розподілених систем. Тому, інша популярна техніка, що може застосовуватись як разом з фрагментацією, так і окремо – це реплікація даних.

Реплікація передбачає створення копій даних (реплік) на кількох вузлах системи, що забезпечує працездатність системи навіть у разі відмови одного або декількох вузлів [5]. Крім того, реплікація може потенційно підвищити загальну продуктивність розподіленої БД, оскільки дозволяє розподілити навантаження між вузлами.

Розрізняють два основні типи – повна та часткова реплікація даних. При повній реплікації кожен з вузлів розподіленої бази даних буде мати повний набір даних та здатен обробляти всі можливі запити користувачів. Недолік цього типу реплікації – підвищені вимоги до сховища вузлів та вищі мережеві витрати для передачі даних всередині системи. Архітектура з повною реплікацією буде менш ефективна в умовах поганої мережевої інфраструктури. Часткова реплікація часто застосовується разом з фрагментацією і в багатьох випадках працює в якості механізму, що здатен підтримувати задану кількість копій/реплік певних фрагментів в системі.

Варто також звернути увагу на різні методи синхронізації даних після початкової реплікації для підтримки системи в узгодженому стані. Розрізняють два методи – синхронна та асинхронна реплікація.

Синхронна реплікація передбачає, що всі зміни, які відбуваються на одному вузлі, негайно застосовуються до всіх інших вузлів розподіленої БД [6]. Вона забезпечує сильну узгодженість (strong consistency) даних, адже зміни вважаються реплікованими (а запит на запис даних в центральний вузол – повністю обробленим) лише після підтвердження оновлення всіх вузлів системи. Синхронна реплікація часто застосовується в класичних реляційних СКБД з підтримкою розподіленого режиму роботи через підтримку сильної узгодженості, яка не заважає підтримці ACID для операцій з даними. Очевидний недолік такого типу синхронізації – це менша швидкодія через потребу узгодження стану всіх вузлів розподіленої бази даних.

Асинхронна реплікація дозволяє основному вузлу (вузлу, який в даний момент функціонує в якості центрального або який прийняв обробку запиту) завершити операцію запису, не чекаючи підтвердження від інших вузлів-реплік [6]. Інші вузли можуть синхронізуватися в будь-який момент з певною затримкою після цього – може використовуватись як push/pull механізми (вузол з оновленими даними надсилає окремі запити на оновлення для інших вузлів або ж вузли самостійно асинхронно перевіряють чи немає нових змін на інших вузла), так й інші більш комплексні системи (event-driven синхронізація, механізм черг тощо). Очевидний недолік асинхронної реплікації – така розподілена БД буде використовувати режим кінцевої узгодженості (eventual consistency), що може не підходити для деяких інформаційних систем (наприклад, банківських систем).

Таким чином, розподілені бази даних активно використовують переваги сучасної мережевої архітектури та покращені апаратні можливості серверів. Через покращені обчислювальні можливості сучасні системи є більш гнучкими й можуть вільно експериментувати зі своєю архітектурою. Застосування різних аспектів таких механізмів, як фрагментація та реплікація даних, дозволяє

побудувати розподілену систему керування базами даних під специфічні сценарії використання з фокусом на найбільш критичні для інформаційної системи характеристики.

Висновок

Було досліджено класифікацію високорівневої архітектури БД за принципами однорідності та різнорідності вузлів. Проведено аналіз відмінностей горизонтальної та вертикальної фрагментації даних, наведено переваги й недоліки статичного та динамічного розподілу фрагментів. Досліджено варіанти застосування повної та часткової реплікації даних, наведено механізми роботи асинхронної та синхронної реплікації. Дослідження показало, що наведені механізми та методи можуть бути важливими елементами високорівневої архітектури розподіленої системи керування базами даних, що значно впливають на кінцеві характеристики системи. З урахуванням цього, подальші напрями розвитку архітектури розподілених СКБД можуть включати в себе спроби покращення та об'єднання цих механізмів, як-от розробка методів динамічного розподілу фрагментів для підвищення швидкодії та масштабованості системи, удосконалення методів реплікації та пошук гібридних підходів для покращення характеристик процесу реплікації, розробка більш ефективних стратегій для розподілу запитів/транзакцій без блокування ресурсів та/або зі збереженням узгодженого стану системи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. The evolution of distributed computing systems: from fundamental to new frontiers [Electronic resource] / Dominic Lindsay [et al.] // Computing. – 2021. – Mode of access: https://doi.org/10.1007/s00607-020-00900-y (date of access: 09.02.2025). – Title from screen.

2. O. O. P. Distributed database management system (DBMS) architectures and distributed data independence [Electronic resource] / Orlunwo Placida O., Prince Oghenekaro A. // International journal of computer science and mobile computing. – 2021. – Vol. 10, no. 1. – P. 23–48. – Mode of access: https://doi.org/10.47760/ijcsmc.2021.v10i01.004 (date of access: 09.02.2025). – Title from screen.

3. Bagui S. Database sharding [Electronic resource] / Sikha Bagui, Loi Tang Nguyen // International journal of cloud applications and computing. – 2015. – Vol. 5, no. 2. – P. 36–52. – Mode of access: https://doi.org/10.4018/ijcac.2015040103 (date of access: 09.02.2025). – Title from screen.

4. Drake M. Understanding database sharding | DigitalOcean [Electronic resource] / Mark Drake // DigitalOcean | Cloud Infrastructure for Developers. – Mode of access: https://www.digitalocean.com/community/tutorials/understanding-database-sharding (date of access: 09.02.2025). – Title from screen.

5. Welekar D. R. Replication data concepts for distributed database systems [Electronic resource] / Dr Rashmi Welekar // Bioscience biotechnology research communications. – 2020. – Vol. 13, no. 14. – P. 344–346. – Mode of access: https://doi.org/10.21786/bbrc/13.14/79 (date of access: 09.02.2025). – Title from screen.

6. Custer C. Synchronous and asynchronous database replication explained (& how data gets lost) [Electronic resource] / Charlie Custer, John Sheaffer // CockroachDB | Distributed SQL for always-on customer experiences. – Mode of access: https://www.cockroachlabs.com/blog/data-loss-prevention-during-outages-you-might-be-losing-data-without-knowing-it/ (date of access: 09.02.2025). – Title from screen.

Миргородський Андрій Вікторович – аспірант групи 121-24а, факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: mirgorodskijav@gmail.com

Романюк Оксана Володимирівна – к.т.н., доцент кафедри програмного забезпечення, Вінницький національний технічний університет, м. Вінниця, e-mail: romaniukoksanav@gmail.com

***Myrhorodskyi Andrii* –** graduate student of group 121-24a, Faculty for Information Technologies and Computer Engineering, Vinnytsia National Technical University, Vinnytsia, e-mail: mirgorodskijav@gmail.com

Oksana Romaniuk – Candidate of Technical Sciences, Associate Professor of the Software Chair, Vinnytsia NationalTechnical University, Vinnytsia, e-mail: romaniukoksanav@gmail.com