УДК 004.658
DOI: 10.31673/2412-9070.2023.052328

В. О. ХОМЕНЧУК, аспірант,
Державний університет інформаційно-комунікаційних технологій, Київ

СТРАТЕГІЇ РОЗПОДІЛУ НАВАНТАЖЕННЯ ДЕЦЕНТРАЛІЗОВАНИХ БАЗ ДАНИХ

Керування навантаженням у базах даних зазвичай охоплює стратегії сегментації, які визначають розподіл та розміщення інформації в системах зберігання даних. У статті розглянуто ключовий аспект сегментації даних та проблеми, пов'язані з вибором правильної стратегії. Деякі із запропонованих стратегій, зокрема стратегія діапазонів та стратегія хешування, мають свої обмеження та ризики, які можуть ускладнювати операції масштабування та переміщення даних. Також зазначено проблеми, пов'язані з вибором конкретного шаблону сегментації в базах даних, включно зі складністю переміщення даних, нерівномірним розподіленням навантаження, проблемами щодо транзакцій та збереження цілісності даних. Вибір правильної стратегії сегментації важливий для забезпечення ефективності та продуктивності системи зберігання даних, тобто він має бути зваженим та відповідати конкретним вимогам системи.

Ключові слова: бази даних; розподілені бази даних; розподілене зберігання даних; СКБД.

ВСТУП

Децентралізовані бази даних є поширеним методом зберігання великої кількості даних. Від вибраного алгоритму розподілу даних між складовими залежить навантаження кожного з елементів, що може призвести до перевантаження одного або кількох вузлів системи.

Аналіз дослідження. Дослідження зосереджено на важливому аспекті керування навантаженням на бази даних. Визначено проблеми, пов'язані з вибором правильної стратегії сегментації, включно з обмеженнями та ризиками, які впливають на операції масштабування та переміщення даних. Важливість правильного вибору стратегії сегментації для забезпечення продуктивності та ефективності системи зберігання даних виокремлюється як ключовий аспект дослідження.

Метою статті є аналіз та дослідження стратегій розподілу навантаження на бази даних.

ОСНОВНА ЧАСТИНА

Для досягнення високої ефективності децентралізованої системи потрібно забезпечити оптимальну продуктивність усіх її компонентів. Вертикальне масштабування, наприклад, за допомогою збільшення ємності дисків, обчислювальної потужності, обсягу пам'яті або кількості мережних адаптерів може тимчасово обійти деякі з обмежень, що виникають у системі. Однак для комерційних застосунків, які підтримують велику кількість користувачів та операцій із величезними обсягами даних, потрібно досягти масштабованості без значних обмежень, тому вертикальне масштабування не завжди є оптимальним рішенням.

Тому в процесі розроблення децентралізованої системи доцільно ретельно розглянути альтернативні підходи до масштабування, зокрема горизонтальне масштабування та інші методи оптимізації. Це дасть змогу забезпечити ефективну роботу системи, яка відповідає вимогам великої кількості користувачів та оптимальному керуванню великими обсягами даних.

Горизонтальне масштабування баз даних має розв'язувати відразу кілька завдань.

• *Масштабування дискового простору*. Сховища даних для великих хмарних застосунків зазвичай містять велику кількість інформації, обсяг якої з часом збільшується. Переважно обсяг дискового простору на сервері обмежений. Хоча є можливість замінювати наявні диски на більш ємнісні або додавати додаткові диски залежно від нагромадження даних. Але потрібно брати до уваги, що рано чи пізно система досягне певної межі, понад яку ємність сховища на цьому сервері збільшити буде неможливо.

• *Обчислювальні ресурси*. Хмарні застосунки мають підтримувати велику кількість користувачів, котрі одночасно працюють, кожен з яких виконує запити для отримання даних зі сховища даних. Один сервер, на якому розміщено сховище даних, може не забезпечити потрібної обчислювальної потужності для підтримання цього навантаження, що призведе до довгого часу відгуку для користувачів та частих збоїв, оскільки застосунки намагатимуться зберігати та отримувати дані. Можливо, можна додати пам'ять або оновити процесори, але система досягне обмеження, якщо більше не вдасться наростити обчислювальні ресурси.

• *Пропускна здатність мережі*. Продуктивність сховища даних, що працює на ізольованому сервері, залежить також від швидкості, з якою сервер може отримувати запити та надсилати відповіді. Коли обсяг мережного трафіку перевищує пропускну здатність мережі, до якої підімкнено сервер, виконання деяких запитів призводить до збоїв — це реальна ситуація.

• *Географія*. Іноді потрібно, щоб створювані користувачами дані зберігалися в тому самому регіоні, де перебувають самі користувачі. Це може бути пов'язано із законодавчими обмеженнями, стандартами або вимогами до продуктивності та припустимої затримки під час доступу до даних. Якщо при цьому користувачі розподілені по різних країнах та регіонах, ви не зможете зберігати всі дані застосунку в одному сховищі даних.

У межах нашого дослідження можливе розділення сховища даних на горизонтальні секції, що дає змогу створювати сегментовану структуру даних. Кожен сегмент має ідентичну структуру даних, проте зберігає різні підмножини інформації. Кожен сегмент є самостійним сховищем даних та може містити інформацію про кілька типів об'єктів. Реалізація сегментованої структури передбачає виконання кожного сегмента на окремому сервері, який використовується як вузол зберігання.

Такий підхід пропонує деякі переваги:

• масштабованість системи через можливість додавання нових сегментів, що працюють на додаткових серверах зберігання;

• зниження вартості й обмежень щодо обладнання завдяки використанню стандартних серверів замість спеціалізованих та витратних комп'ютерів для кожного вузла зберігання;

• мінімізація конфліктів та підвищення продуктивності завдяки раціональному розподілу робочого навантаження між сегментами;

• оптимізація доступу до даних завдяки фізичному розміщенню сегментів у хмарних середовищах ближче до кінцевих користувачів, які взаємодіють із цими даними.

Для досягнення оптимального балансу між продуктивністю та масштабованістю вкрай важливо в належний спосіб розподілити дані відповідно до типів запитів, що виконує застосунок. У багатьох випадках схеми сегментування не зможуть точно відповідати вимогам кожного окремого запиту. Наприклад, в мультитенантній системі застосунку може бути потрібним отримати дані клієнта за його ідентифікатором, а також здійснити пошук цих даних за іншими атрибутами, зокрема ім'ям чи місцезнаходженням клієнта. У такому разі бажано використовувати стратегію сегментування, де ключ сегмента відповідає переважно виконуваним запитам.

Якщо потрібно регулярно отримувати дані, використовуючи комбінацію значень деяких атрибутів, можна визначити композитний ключ сегмента, що зв'язує необхідні атрибути. Також можна застосовувати інший підхід — використовувати шаблон таблиці індексів, який дає змогу швидко відшукувати дані за значеннями атрибутів, які не належать до ключа сегмента.

Під час вибору ключа сегмента та правил розподілу даних за сегментами зазвичай використовується одна з трьох стратегій. Варто зауважити, що між сегментами та серверами, на яких вони розміщені, не обов'язково має бути відповідність «один до одного», один сервер може містити кілька сегментів.

Стратегія пошуку. У цьому варіанті логіка сегментування використовує карту відповідності за ключем сегмента та направляє запити до того сегмента, що містить необхідні дані. Наприклад, у багатокористувацькому застосунку всі дані кожного клієнта можна зберігати в одному сегменті, тоді ідентифікатор клієнта можна використовувати як ключ сегмента. Дані кількох клієнтів можуть зберігатися в одному сегменті, але дані одного клієнта не можуть опинитися в кількох сегментах. Варіант сегментування даних за ідентифікаторами клієнтів зображено на рис. 1.

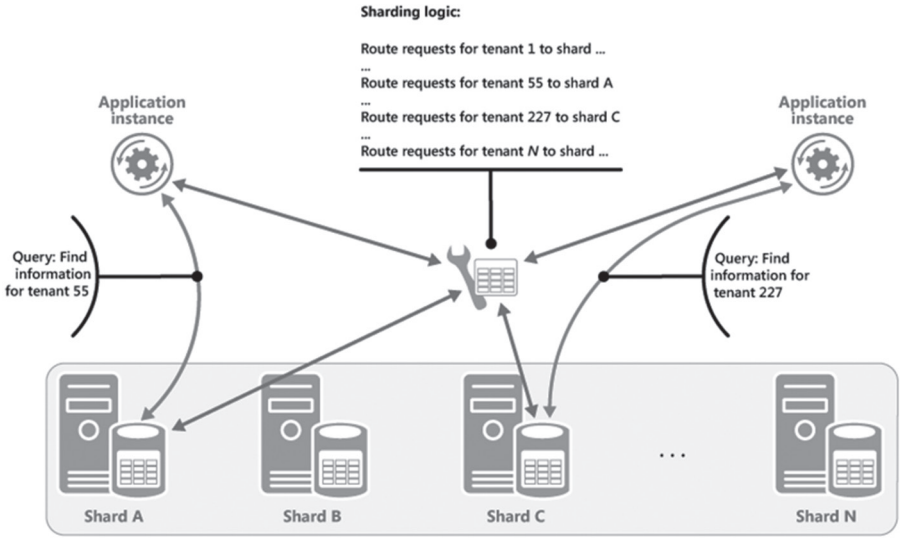

Рис. 1. Сегментування даних за ідентифікаторами клієнтів

Для зіставлення значень ключа сегмента та фізичного сховища можна використовувати фізичні сегменти, тоді кожен ключ сегмента буде зіставлений із фізичною секцією. Існує і більш гнучкий спосіб розподілу сегментів — віртуальне секціонування. У цьому разі ключі сегмента зіставляються з такою самою кількістю віртуальних сегментів, які зі свого боку зіставляються з меншою кількістю фізичних секцій. У такому варіанті застосунок відшукує дані за допомогою ключа сегмента, який посилається на віртуальний сегмент, і система автоматично зіставляє віртуальні сегменти з фізичними секціями. Таке зіставлення між віртуальними сегментами та фізичними секціями можна змінити в будь-який момент, не вдаючись до програмних методів, а просто застосувавши інший набір ключів сегмента.

Стратегія діапазонів. Ця стратегія полягає в групуванні пов'язаних елементів в одному сегменті та їх упорядкуванні за ключем сегмента (ключі сегментів є послідовними). Це корисно для застосунків, які часто отримують набори елементів з деякого діапазону значень (запити до даних повертають набір елементів, для яких значення ключа сегмента потрапляють у заданий діапазон). Наприклад, якщо застосунку регулярно потрібно отримувати список усіх замовлень, розміщених у певний місяць, ці дані можна здобувати швидше, розмістивши замовлення за кожний місяць в одному сегменті та упорядкувавши їх за датою та часом. Якщо замовлення розташовані в різних сегментах, їх доведеться витягати окремо, тобто виконувати велику кількість окремих запитів (що повертає один елемент даних). Схему зберігання послідовних наборів (діапазонів) даних у сегментах наведено на рис. 2.

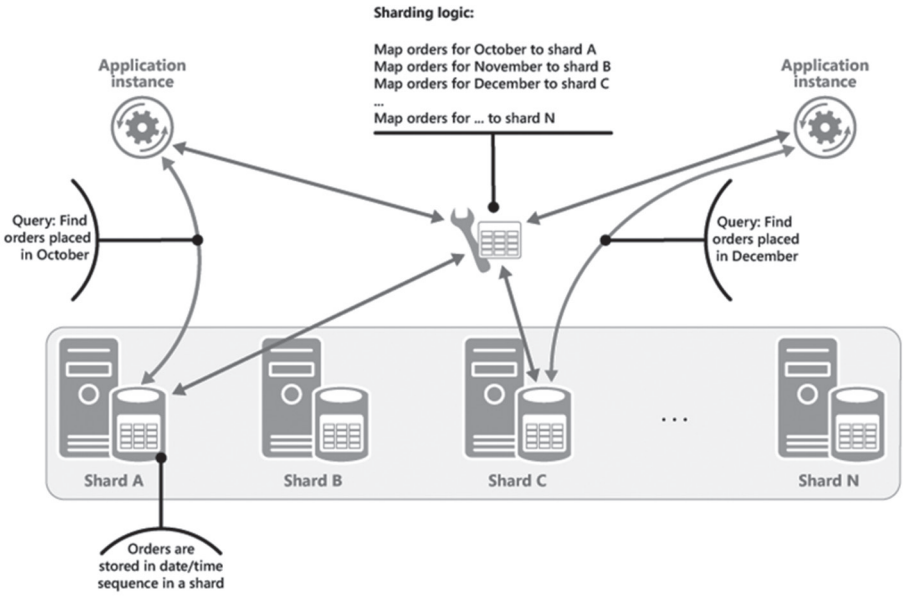

Рис. 2. Сегментування даних за діапазонами

У цьому прикладі використовується складний ключ сегмента, в якому найбільш важливим елементом є місяць замовлення, а за ним йдуть день та час замовлення. У разі створення нових замовлень та розміщення їх за сегментами дані натуральним чином сортуються. Деякі сховища даних підтримують ключі сегмента, які складаються з двох елементів: ключа секції, що визначає сегмент, і ключа рядка, що однозначно визначає елемент у межах сегмента. Зазвичай дані в сегменті зберігаються в порядку значень ключа рядка. Елементи, для яких застосовуються запити за діапазонами значень і які потрібно зберігати в одному сегменті, матимуть однакові значення ключа секції та унікальні значення ключа рядка.

Стратегія хешування. Ця стратегія дає можливість знизити ймовірність виникнення високоактивних сегментів (сегментів із надмірно високим навантаженням). У цьому варіанті дані розподіляються між сегментами так, щоб дотримувалась рівновага між розміром кожного сегмента та середнім навантаженням, що припадає на цей сегмент. Логіка сегментації визначає, в якому сегменті зберігати кожний елемент даних за значенням хеш-індексу за одним або кількома атрибутами даних.

Функцію хешування має бути вибрано в такий спосіб, щоб рівномірно розподілити дані між сегментами. Можливо, для цього доведеться додати до формули обчислення деякий випадковий елемент. Сегментування даних за хешем ідентифікаторів клієнтів унаочнює рис. 3.

Щоб зрозуміти переваги стратегії хешування порівняно з іншими стратегіями сегментації, розглянемо такий приклад. Багатокористувацький застосунок по черзі реєструє нових клієнтів та розподіляє здобуті дані між сегментами у сховищі даних. Якщо використовувати стратегію діапазонів, всі дані про клієнтів з ідентифікаторами від 1 до n зберігаються в сегменті A, з ідентифікаторами від n+1 до m

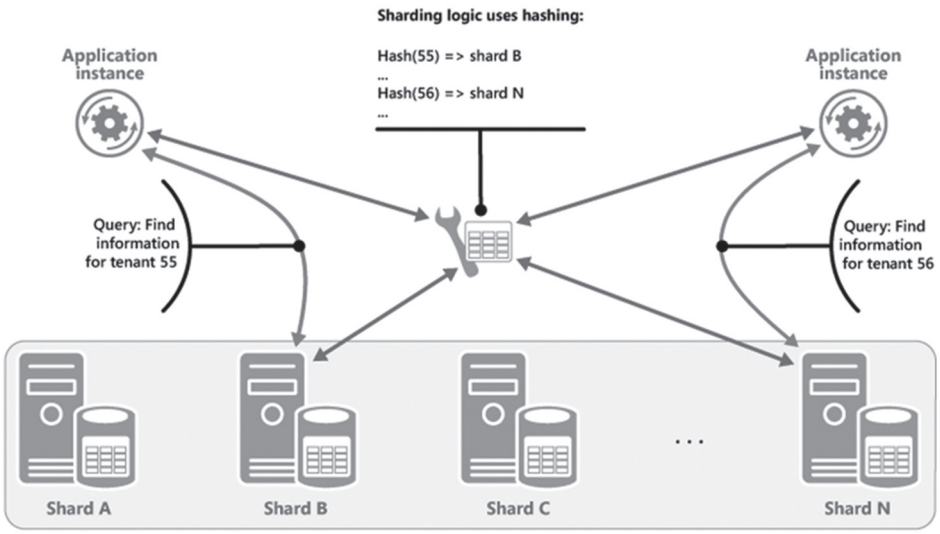

Рис. 3. Сегментування даних за хешем ідентифікаторів клієнтів

— у сегменті *B* і т. д. Якщо припустити, що найбільшу активність проявляють недавно зареєстровані клієнти, максимальне навантаження з роботи з даними припаде на невелику кількість сегментів, що створить високоактивний сегмент. У цій самій ситуації стратегія хешування буде розподіляти клієнтів за сегментами на основі хешу ідентифікатора. Це означає, що клієнти з послідовними ідентифікаторами, імовірно, потраплять у різні сегменти, що дасть змогу рівномірно розподілити навантаження. Саме таку поведінку для клієнтів з ідентифікаторами 55 та 56 зображено на рис. 3.

Переваги та особливості кожної з трьох основних стратегій сегментування

Стратегія пошуку. Ця стратегія надає максимальний контроль над налаштуванням та використанням сегментів. Застосування віртуальних сегментів дає змогу усунути проблеми з балансуванням даних завдяки можливості додавати нові фізичні секції для розподілу робочого навантаження. Відповідність віртуальних сегментів та фізичних секцій, на яких їх реалізовано, можна змінювати, не застосовуючи програмні методи, тобто не змінюючи код застосунку, який зберігає та отримує дані за значенням ключа сегмента. Пошук розташування сегмента може створювати додаткове навантаження.

Стратегія діапазонів. Легко реалізовується та добре працює із запитами за діапазонами, оскільки часто здатна за допомогою однієї операції дістати кілька елементів даних з одного сегмента. Ця стратегія спрощує процеси керування даними. Наприклад, якщо охопити в одному сегменті користувачів з одного регіону, можна запланувати оновлення окремо для кожного часового поясу на основі локальних особливостей використання навантаження та запитів. Але ця стратегія не дає змоги забезпечити оптимальне балансування між сегментами. Перерозподіл даних між сегментами буде складним і не розв'яже проблеми нерівномірного навантаження, якщо основна частина навантаження створюється для даних із близькими значеннями ключа сегмента.

Стратегія хешування. Уможливлює розподілення даних та навантаження найбільш рівномірно. Маршрутизацію запитів виконує прямо із застосунку за допомогою хеш-функції. Немає потреби використовувати карту відповідності. Обчислення хешу може створити додаткове навантаження. Крім того, це ускладнить перерозподіл навантаження між сегментами.

Операції масштабування та переміщення даних

Кожна стратегія сегментації створює різні можливості та різні рівні складності в контексті зміни масштабу, переміщення даних та підтримання стану системи.

Стратегія пошуку дає змогу виконувати масштабування та переміщення даних на рівні користувачів, і для цього не потрібно вимикати систему. Достатньо тимчасово призупинити операції користувачів або їхню частину (це можна зробити в періоди низького навантаження), потім перемістити дані в інший віртуальний сегмент або інший фізичний сегмент, змінити карту відповідностей та оголосити недійсними або оновити всі кешовані дані. Після цього можна відновити роботу користувачів. Операції такого типу часто можна виконувати централізовано. Стратегія пошуку потребує узгодженості даних із кешуванням та реплікацією.

Стратегія діапазонів накладає деякі обмеження на операції масштабування та переміщення даних. Зазвичай їх слід виконувати, від'єднавши від мережі сховище даних або його частину, оскільки дані

потрібно ділити та об'єднувати в багатьох сегментах. Переміщення даних для балансування сегментів не завжди дає змогу вирішити проблему нерівномірного навантаження, якщо основна частина навантаження створюється для даних із близькими значеннями ключа сегмента або ідентифікаторами з одного діапазону. Стратегія діапазонів також може охоплювати дії з підтримання стану, щоб діапазони можна було відповідним чином зіставити з фізичними секціями.

Стратегія хешування значно ускладнює операції масштабування та переміщення даних, оскільки ключі секцій — це хеш-індекси ключів сегментів або ідентифікаторів даних. Потрібно визначити нове розташування для кожного сегмента на основі результатів виконання хеш-функції або змінити цю функцію для правильної відповідності. Однак стратегія хешування не потребує підтримання стану.

Проблеми шаблонів сегментації баз даних

• Складність і ризики переміщення даних. У процесі використання деяких стратегій сегментування, зокрема стратегії діапазонів або хешування, переміщення даних в інший сегмент або сервер може бути складним та ризикованим. Це може призвести до тимчасової недоступності даних або втрати їх цілісності під час процесу переміщення.

• Нерівномірне навантаження. Деякі шаблони сегментації можуть призводити до нерівномірного розподілу навантаження між сегментами бази даних. Це може спричинити перевантаження деяких сегментів, тоді як інші залишатимуться недостатньо використаними, що може погіршити продуктивність та завдати шкоди користувачам.

• Поділ транзакцій. Під час використання деяких стратегій сегментування, наприклад, коли дані розподілені між різними фізичними серверами, виникає проблема поділу транзакцій між сегментами. Це може збільшити складність керування цілісністю даних та збільшити ймовірність виникнення проблем з одночасним доступом до даних.

• Перевищення обсягу даних. Іноді стратегії сегментування можуть зазнавати проблем з обмеженням максимального обсягу даних, який може бути збережений у кожному сегменті. Це може створювати обмеження щодо розміру бази даних, що може змінюватись із часом і приводити до потреби в переструктуризації сегментів.

• Складність керування станом. У разі використання деяких стратегій сегментування, зокрема стратегії діапазонів, потрібно відстежувати та керувати станом сегментів та їх відповідностями до фізичних серверів. Це може бути викликом під час масштабування та підтримання системи.

ВИСНОВОК

Вибір конкретного шаблону сегментування баз даних має здійснюватись з огляду на особливості конкретної системи та вимоги до даних, а також має бути підтриманим належним контролем, щоб уникнути проблем у майбутньому.

Список використаної літератури

*1. **Avraham Leff, Philip S. Yu.** An adaptive strategy for load sharing in distributed database environments with information lags // Journal of Parallel and Distributed Computing. 1991. Vol. 13, Issue 1.*

*2. **An adaptive** multi-level caching strategy for Distributed Database System / Feng Lu, Ziqian Shi, Lin Gu, Hai Jin, Laurence Tianruo Yang // Future Generation Computer Systems. 2019. Vol. 97. P. 61–68.*

*3. **A Query** Optimization Strategy for Autonomous Distributed Database Systems World Academy of Science, Engineering and Technology / I. Dina, S. Elsayed, Kh. Dina, H. Dina // International Journal of Computer and Information Engineering. 2018. 12. 10.1999/1307-6892/10008664.*

V. O. Khomenchuk

LOAD DISTRIBUTION STRATEGY OF DECENTRALIZED DATABASES

Load management in databases typically involves segmentation strategies that determine the distribution and placement of information within data storage systems. This article explores a critical aspect of data segmentation and the challenges associated with selecting the right strategy. Some of the discussed strategies, such as range-based strategy and hashing strategy, come with their limitations and risks that can complicate scalability and data migration operations. The article also highlights issues related to choosing a specific data segmentation pattern in databases, including data movement complexity, uneven load distribution, transactional problems, and data integrity preservation. Choosing the right segmentation strategy is crucial to ensure the efficiency and productivity of a data storage system, and it should be a thoughtful choice tailored to the specific requirements of the system. The article discusses important data segmentation strategies in storage systems, which play a key role in the distribution and management of large datasets.

The authors explore three main strategies: the search strategy, the range strategy, and the hashing strategy. Each of these strategies has its unique advantages and characteristics. The search strategy provides maximum control over data placement and the ability to use virtual segments for data balancing. This strategy is particularly useful for multi-user applications but may introduce additional overhead during segment retrieval. The range strategy groups related data into one segment and works well with queries for ranges of values. However, it can lead to uneven load distribution. The hashing strategy helps evenly distribute the load and avoid highly active segments but may introduce additional computational overhead when calculating the hash. The article also discusses data scaling and relocation operations for each of these strategies. For example, the search strategy allows for data scaling and relocation at the user level without system downtime. This article helps readers better understand the differences between data segmentation strategies and choose the most suitable one for their specific use case.

Keywords: databases; distributed databases; distributed data storage; DBMS.