

ПІДВИЩЕННЯ ШВИДКОСТІ ВИКОНАННЯ SQL-ЗАПИТІВ ШЛЯХОМ ВИКОРИСТАННЯ ПАРТИЦІЮВАНЬ НА РІВНІ СКБД

Чопенко М.С.¹; Гайда А.Ю., доц., к.т.н.²

^{1,2} Національний університет кораблебудування імені адмірала Макарова

^{1,2} Україна, Миколаїв

¹ webnikitchopenko@gmail.com; ² cetus@ukr.net

Анотація. Ефективність сучасних інформаційних систем визначається можливостями накопичення значних обсягів даних і їх обробки за прийнятні для користувачів терміни. Істотною проблемою створення таких систем є забезпечення стабільних експлуатаційних характеристик в умовах зростання обсягів даних, ускладнення структури даних та запитів на їх отримання. В роботі розглянуті механізми підвищення швидкості доступу до даних РСКБД, що базуються на партиціюванні таблиць даних та визначені критерії партиціювання залежно від обсягів даних, що визначаються запитом до бази даних. Визначено, що раціональне використання механізмів партиціювання дозволяє істотно зменшити час обробки запитів і підвищує продуктивність бази даних.

Ключові слова: система керування базами даних; реляційна база даних; Structured Query Language; PostgreSQL; партиції.

Вступна частина. У сучасних високонавантажених системах ефективність бази даних (БД) відіграє ключову роль в процесах збереження і обробки інформації. Однією з найбільш поширених технологій для організації і управління даними є реляційні системи керування базами даних (РСКБД), а основним засобом звернення до СКБД є SQL-запити. Зі збільшенням обсягів даних в базі даних і, головне, складності структури бази даних та підвищенні вимог щодо первинної обробки даних на рівні СКБД зростає складність SQL-запитів до БД. Як наслідок, швидкість доступу до даних може суттєво зменшуватись та впливати на загальне функціонування системи. Тривалі запити не тільки споживають значну кількість системних ресурсів, що може спричинити перевантаження сервісів, але також здатні викликати тимчасове блокування таблиць та, навіть, призвести до втрати або пошкодження даних на рівні додатків. Популярним шляхом вирішення зазначеної проблеми є підвищення вимог до апаратної частини, але це не завжди дає очікуваний результат.

Загалом, на поточний момент визначено основні проблеми, які зустрічаються при роботі з великими обсягами даних в БД:

- Відсутність індексів або їх неефективне використання. Індеси дозволяють прискорювати пошук по таблицях, але значна складність структур даних і запитів може призводити до необхідності створення надмірної кількості індексів, що призводить до розростання бази даних та збільшенням вимог до обчислювальних ресурсів.
- Значне перевищення обсягів даних у БД над обсягами вибірки даних у конкретних запитах. Партиціювання таблиць дозволяє розділити великі таблиці на менші частини за певним критерієм, наприклад, за датою або за діапазонами значень, що дозволяє СКБД виконувати запити над окремими партиціями БД, а не БД у цілому.
- Нераціональна або недостатня конфігурація серверів бази даних. Параметри СКБД, такі як розмір буферів, кешування даних в оперативній пам'яті, які не відповідають

потребам конфігурації БД та складності запитів, призводять до уповільнення обробки запитів.

- Невідповідність апаратної частини СКБД обсягам даних та складності запитів. Недостатня оперативна пам'ять, перевантаження процесорів через численні одночасні запити може призвести до уповільнення виконання операцій. В таких випадках краще використовувати шардування. Це дозволить розподілити дані між кількома серверами для горизонтального масштабування, для підвищення продуктивності та балансування навантаження.
- Недостатня або надмірна нормалізація даних БД. Якщо база даних має не оптимізовану структуру, це може викликати затримки у виконанні запитів. Надмірна кількість зв'язків між таблицями або використання не відповідних типів даних негативно впливають на продуктивність SQL.

Отже, на поточний момент є багато способів збільшення швидкості виконання запитів – це використання індексів, оптимізованих структур даних, шардування та партиціювання даних, алгоритми з'єднання та обмеження вибірки. Ці засоби частково впроваджені в СКБД їх розробниками і також можуть впроваджуватись проєктантами БД. Для створення ефективних SQL-запитів найчастіше використовуваним способом є створення декількох варіантів запиту та отримання плану їх виконання для подальшого аналізу та визначення оптимальних варіантів, а також правильне використання вбудованих інструментів.

Дослідження в області оптимізації бази даних підкреслюють важливість використання партиціювання для високонавантажених продуктивних систем.

Автор Танаєма П. [1] та інші стверджують, що продуктивність бази даних є одним із основних компонентів підтримки стійкості системи. У дослідженні вони провели проектування та оцінку таблиць розділів, яка потім почала використовуватись у продуктовому середовищі. Результати показали, що таблиці з партиціями мають кращу продуктивність, але наголосили, що вставка працює погано порівняно зі звичайними таблицями.

Науковець Беднарчук П. [2] показав, що поширеною проблемою доступу до БД є уповільнення звітів у базах даних з часом. На початку система працює швидко та ефективно, але з часом вона сповільнюється і без впровадження відповідних рішень система може перестати відповідати вимогам. Дослідник довів, що завдяки партиціюванню можливо досягти однакової продуктивності системи за весь період експлуатації, не зважаючи на збільшення обсягів накопичених даних та ускладнення запитів до БД.

Автор Мустафа С.[3] та інші вважають, що багато факторів впливають на рішення щодо розділення даних на партиції. Це розмір бази даних, тип даних, тип запитів та частота запитів. Дослідження показали, що абсолютного ультиматуму не існує, а конкретне рішення потрібно вибирати емпіричним шляхом.

З огляду на дослідження різних науковців можна зробити висновок, що партиціювання є одним із ключових методів оптимізації продуктивності баз даних, особливо в умовах високого навантаження та роботи з великими обсягами інформації.

Метою дослідження є збільшення швидкості та ефективності виконання SQL-запитів шляхом використання партиціювань на рівні СКБД.

Основна частина. Розподілення таблиць SQL – це процес розподілу даних таблиці на кілька таблиць. Це дозволяє покращити продуктивність запитів під час сканування даних з одного або декількох розділів, а не при читанні чи пошуку по всій таблиці (або її індексу). Якщо фільтр запиту охоплює одну або декілька партицій, багато з них виключається зі сканування. Загальна схема партиціювання таблиці зображена на рис 1.

В PostgreSQL є три типи декларативного партиціювання таблиці: розбиття по діапазону значень, розподіл списку, розподілення хешу, це зображено на рис 2. Розбиття за діапазоном дозволяє ділити таблицю на частини на основі діапазонів значень одного або кількох стовпців. За списком на основі конкретних значень стовпця, де кожна партія містить записи, що відповідають певному набору значень. Розбиття за хешем дозволяє розподілити значення на основі хеш функції стовпця. В нашому випадку ми використовуємо розбиття даних за діапазоном [4].

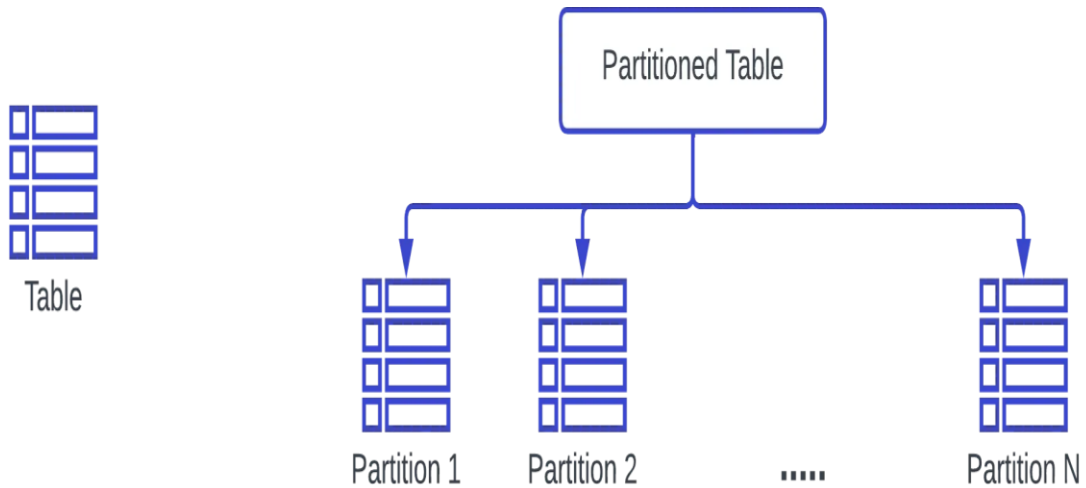


Рис 1. Загальна схема партиціювання таблиці

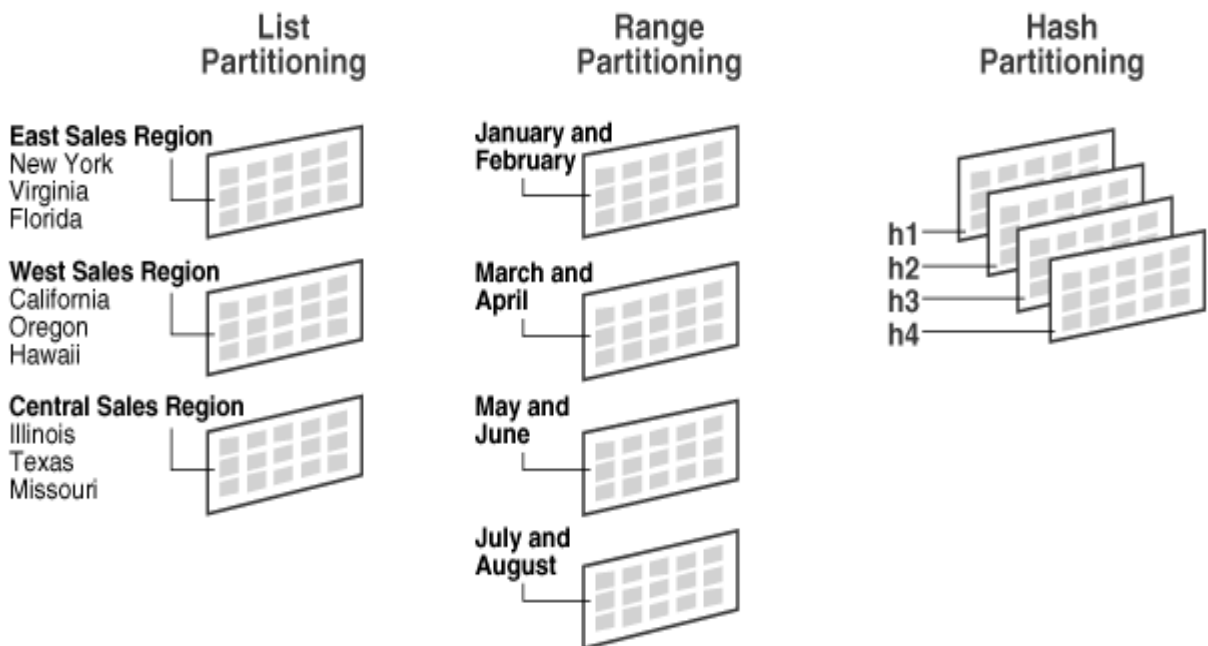


Рис 2. Типи партиціювання таблиці

Проведемо експеримент, який досліджує вплив кількості партицій таблиці на продуктивність операції вибору даних. Було вибрано реляційну базу даних PostgreSQL. Характеристики комп'ютера, на якому проводяться експерименти, описані у таблиці 1.

Таблиця 1 – Опис комп'ютера, що використовується

Характеристика	Значення
Процесор	Десяти ядерний 2021 M1 PRO
Оперативна пам'ять	16 ГБ
Операційна система	MacOS Ventura 13.0
Диск	SSD 256 gb, запис та читання до 3,3 ГБ/с

Було згенеровано таблицю користувачів, яка імітує систему з 60 мільйонами записів та має рівномірний розподіл по 5 мільйонів за датою реєстрації протягом 2023 року. У ході виконання досліджень застосуємо діапазон значень у якості логічного розділення даних для поля дата реєстрації. За допомогою команди EXPLAIN ANALYZE ми можемо побачити час виконання запитів з використанням партицій.

Хід виконання:

1. Створимо таблицю користувачів: `CREATE TABLE users_no_partition (id INT, first_name VARCHAR(25) NOT NULL, date_joined DATE NOT NULL) PARTITION BY RANGE(date_joined).`
2. Заповнимо 60 млн. користувачів з датою реєстрації 2023 року: `INSERT INTO users_no_partition (id, first_name, date_joined) VALUES(GENERATE_SERIES(1, 5000000), 'Bob',(TIMESTAMP '2023-01-01' + RANDOM() * (TIMESTAMP '2023-01-31' - TIMESTAMP '2023-01-01'))::DATE.`
3. Створимо партиції для таблиці, з різним діапазоном значень для поля дата реєстрація користувача. Приклад запиту з квартальним діапазоном: `CREATE TABLE users PARTITION of users for VALUES FROM ('2023-01-01') to ('2023-04-01').`
4. Виконуємо запит з пошуком користувачів, які зареєструвались у грудні, не змінюючи дати пошуку для кожного діапазону розбиття таблиці: `SELECT COUNT(*) FROM users_no_partition WHERE date_joined >= '2023-12-01' AND date_joined <= '2023-12-31'.`
5. Повторюємо запит пошуку для різних діапазонів розбиття: місяць, 3 місяці, квартал, півроку.

Результати виконання запитів представлені в таблиці 2 та на рисунку 3.

Таблиця 2 – Час виконання запиту вибору користувачів з різним діапазоном розбиття партицій

Діапазон розбиття	Значення
Відсутній	1500 мілісекунд
Тиждень	470 мілісекунд
Місяць	160 мілісекунд
3 Місяці	636 мілісекунд
Квартал	811 мілісекунд
Півроку	1266 мілісекунд

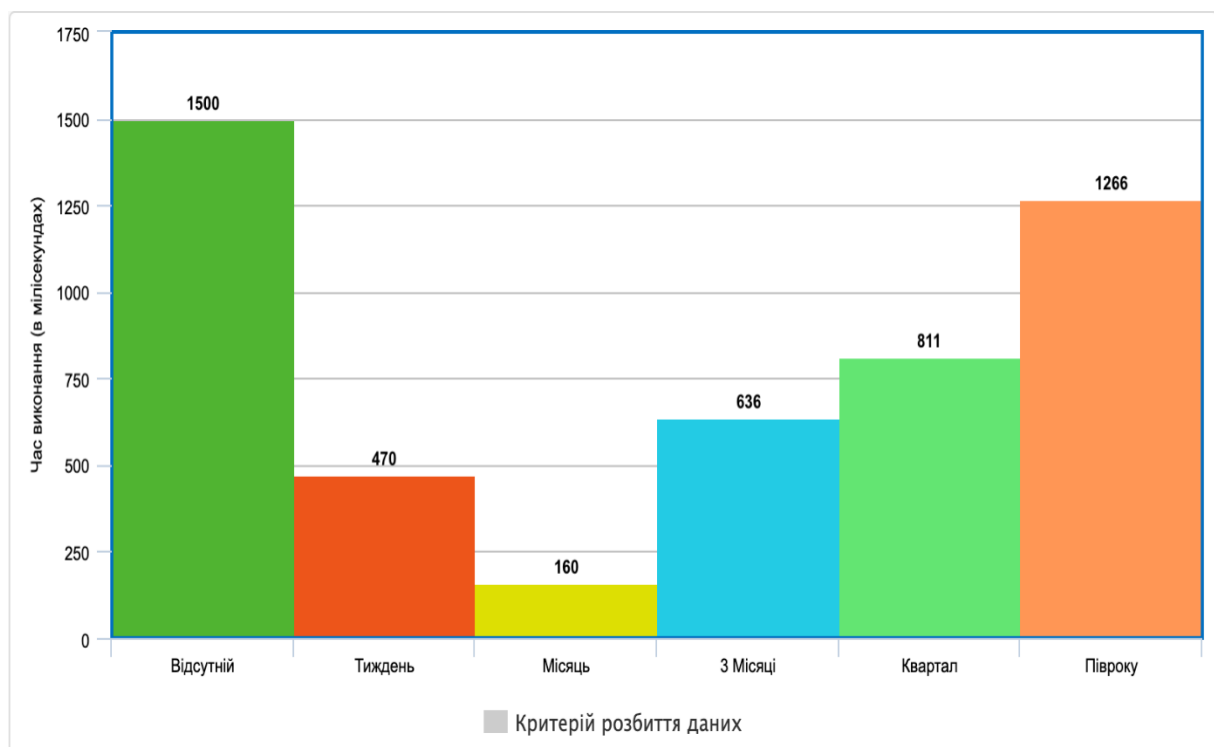


Рис 3. Діаграма виконання запитів з критеріями розбиття даних і відповідним часом виконання

Партиціювання за місяцями забезпечило найкращу продуктивність, скоротивши час виконання на 89,3% порівняно з відсутністю партицій. Партиція за 3 місяці скоротила час на 57,6%, а за квартал на 45,9%. Найгірший результат дала партиція за півроку, яка скоротила час виконання лише на 15,6%, що вказує на значне зниження ефективності при великих інтервалах партиціювання. Розбиття таблиць на партиції за тижнями є повільнішим порівняно з розбиттям за місяцями, що зумовлено збільшенням кількості партицій і, як наслідок, зростанням витрат на їх обробку.

Висновки. Спочатку система може функціонувати швидко та ефективно, але зі збільшенням даних її швидкодія погіршується. У ході дослідження було проведено аналіз ефективності партиціювання таблиць для підвищення швидкості виконання SQL-запитів. Результати експериментів на прикладі великої таблиці користувачів показали, що партиціювання за короткі інтервали, зокрема за місяцями, забезпечує найбільше скорочення часу виконання запитів. Для досягнення найкращих результатів зменшення часу виконання SQL необхідно експериментувати з критерієм розподілу даних. Таким чином, дослідження

підтвердило, що правильний вибір інтервалу партиціювання може суттєво покращити швидкість доступу до даних і підвищити загальну ефективність бази даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] Tanaem, P. F., Tanaamah, A. R., & boymau, infrain oktofianus. (2022a). *Table Partition in Stars: Concept and Evaluations*. <https://doi.org/10.36227/techrxiv.21324264>
- [2] Bednarczuk, P. (2020). Optimization in very large databases by partitioning tables. *Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska*, 10(3), 95–98. <https://doi.org/10.35784/iapgos.2056>
- [3] Matalqa, S., & Mustafa, S. (2016). The Effect of Horizontal Database Table Partitioning on Query Performance . *The International Arab Journal of Information Technology*, 13, 184–189.
- [4] Schönig, H.-J. (2019). *Mastering postgresql 12: Advanced techniques to build and administer scalable and reliable postgresql database applications*, 3rd Edition. Packt Publishing, Limited.

Chopenko M. S. , Gaida A. Y.

Improving SQL Query Execution Speed Through the Use of Partitioning at the DBMS Level

Abstract. The efficiency of modern information systems is determined by their ability to accumulate significant volumes of data and process it within acceptable timeframes for users. A critical challenge in creating such systems is ensuring stable operational characteristics in the face of growing data volumes and the increasing complexity of data structures and queries for data retrieval. This work examines mechanisms for improving data access speed in relational database management systems (RDBMS) based on table partitioning and defines partitioning criteria depending on the data volumes specified by the queries to the database. It has been established that the rational use of partitioning mechanisms significantly reduces query processing time and enhances database performance.

Keywords: Database Management System; relational database; Structured Query Language; PostgreSQL; partitions.