УДК 004.9

Голубінка В.П.[1]

МЕТОДИ СИСТЕМНОГО АНАЛІЗУ У ЗАДАЧАХ ОПТИМІЗАЦІЇ СКЛАДНИХ SQL-ЗАПИТІВ

Оптимізація SQL-запитів є важливим напрямом у сфері управління базами даних, оскільки ефективність обробки складних запитів безпосередньо впливає на швидкість роботи інформаційних систем. У сучасних умовах обробки великих обсягів даних необхідно не лише правильно формулювати запити, а й застосовувати методи системного аналізу для їх оптимізації. Використання системного підходу дозволяє оцінити продуктивність запитів, визначити їх слабкі місця та знайти способи підвищення ефективності виконання.

У сучасних умовах, коли обробка великих обсягів даних є критично важливою, системний аналіз дозволяє оцінити продуктивність SQL-запитів, визначити найкритичніші операції та запропонувати методи їхньої оптимізації. Дослідження в цій галузі охоплюють як теоретичні підходи до аналізу запитів, так і практичні інструменти для їхнього профілювання та оптимізації.

Дослідження питань оптимізації SQL-запитів та використання методів системного аналізу у сфері управління базами даних широко представлені в науковій літературі. Значний внесок у розвиток теоретичних основ оптимізації запитів зробили американські вчені Авраам Сілбершац, Генрі Корт та Сударшан, які у своїй праці *Database System Concepts* детально розглянули механізми роботи оптимізаторів запитів та принципи їхньої ефективної побудови [2]. Крім того, Рагурам Елмасрі та Шамкант Наватхе у підручнику *Fundamentals of Database Systems* описали принципи побудови реляційних баз даних та їхньої оптимізації через застосування реляційної алгебри та аналізу виконання запитів [1]. Дослідження алгоритмів обробки запитів також проводили Джеффрі Ульман та Гектор Гарсія-Моліна, які у своїй книзі *Database Systems: The Complete Book* приділили увагу питанням індексації, використання хеш-з'єднань та багаторівневої оптимізації запитів [5]. У сфері практичної оптимізації баз даних значну увагу приділяють аналітики, які працюють із PostgreSQL та Oracle, зокрема в офіційній документації PostgreSQL Global Development Group розглянуто принципи роботи планувальника запитів та алгоритми вибору

[1] аспірант кафедри ІСМ, НУ «Львівська політехніка»

оптимального плану виконання [3]. Дослідження в цій сфері тривають, і особливої уваги потребує питання автоматизації процесів оптимізації через застосування штучного інтелекту та машинного навчання.

Процес оптимізації SQL-запитів ґрунтується на кількох фундаментальних принципах, кожен з яких спрямований на підвищення ефективності обробки даних у системах управління базами даних. Перший із них полягає в мінімізації обчислювальних витрат, що досягається завдяки застосуванню оптимізованих алгоритмів зчитування та обробки інформації, що, у свою чергу, сприяє зменшенню часу виконання запитів. Другий принцип передбачає раціональне використання апаратних ресурсів, включаючи процесорний час, оперативну пам'ять та дисковий простір. Його реалізація базується на детальному аналізі навантаження системи з метою запобігання надмірному використанню ресурсів та забезпечення їх збалансованого розподілу. Третій принцип орієнтований на зниження обсягу переданих даних, що реалізується шляхом ефективного використання індексів, агрегатних функцій та механізмів кешування проміжних результатів.

З метою практичної реалізації зазначених підходів у процесі оптимізації SQL-запитів застосовуються методи системного аналізу, які дозволяють оцінити логічну структуру запиту, проаналізувати його план виконання та ідентифікувати потенційні точки неефективності. Використання цих методів дає змогу здійснювати глибоку діагностику процесів обробки даних та забезпечувати їх раціональне виконання в межах сучасних інформаційних систем.

Одним із найважливіших інструментів для аналізу SQL-запитів є команда EXPLAIN, яка дозволяє отримати детальну інформацію про те, як система управління базами даних (СУБД) виконує запит. Наприклад, у PostgreSQL команда EXPLAIN ANALYZE дає можливість побачити реальні витрати часу на виконання кожного етапу запиту. Аналізуючи план виконання, можна виявити використання послідовного сканування (Seq Scan), яке зазвичай є менш ефективним, і замінити його на використання індексованого сканування (Index Scan), що значно пришвидшує вибірку даних [3].

Одним із найбільш ефективних методів оптимізації є використання індексів. Індекси дозволяють значно зменшити кількість операцій пошуку в таблиці, що особливо важливо для великих баз даних. B-Tree індекси є найпоширенішими та використовуються для швидкого пошуку, тоді як Hash-індекси ефективні для рівнозначного порівняння, але не підтримують сортування. Крім того, існують GIN та GiST-індекси, які застосовуються для складних типів даних, таких як текстовий пошук [2].

Ще одним важливим аспектом оптимізації SQL-запитів є правильний вибір стратегії з'єднання таблиць. Використання JOIN є однією з найбільш ресурсомістких операцій, тому необхідно вибирати оптимальний алгоритм з'єднання. Наприклад, Nested Loop Join є ефективним для малих таблиць, але може бути повільним на великих наборах даних. Hash Join, навпаки, добре працює для великих таблиць, але потребує додаткової пам'яті. Merge Join є ефективним, якщо обидві таблиці відсортовані за ключовим полем [1].

Ще одним методом оптимізації є кешування результатів SQL-запитів. Це дозволяє зменшити кількість повторних звернень до бази даних і підвищити продуктивність системи. Для цього можуть використовуватися матеріалізовані представлення (Materialized Views), збережені процедури (Stored Procedures) або зовнішні системи кешування, такі як Redis або Memcached [4].

Процес оптимізації SQL-запиту включає кілька етапів. На першому етапі проводиться аналіз вихідного запиту за допомогою команди EXPLAIN ANALYZE, що дозволяє виявити вузькі місця. Далі здійснюється вибір оптимального методу JOIN та перевіряється ефективність використання індексів. Після цього проводиться тестування різних варіантів запиту та обирається найефективніший. Завершальним етапом є впровадження кешування або використання матеріалізованих представлень для зменшення навантаження на базу даних [5].

Методи системного аналізу відіграють важливу роль у процесі оптимізації SQL-запитів, дозволяючи суттєво підвищити ефективність роботи баз даних. Комплексний підхід до оптимізації включає використання індексів, правильне структурування запитів, аналіз планів виконання, а також застосування сучасних алгоритмів і технологій машинного навчання. Подальші дослідження можуть бути зосереджені на автоматизації процесів оптимізації та впровадженні інтелектуальних систем для підвищення продуктивності обробки складних SQL-запитів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Elmasri R., Navathe S. B. Fundamentals of database systems. 7th ed. Pearson, 2020. 1272 p.
2. Molina H. G., Ullman J. D., Widom J. Database systems: The complete book. 2nd ed. Pearson, 2019. 1248 p.
3. Silberschatz A., Korth H. F., Sudarshan S. Database system concepts. 7th ed. McGraw-Hill, 2019. 1376 p.
4. PostgreSQL query planner and optimizer. *PostgreSQL Global Development Group*. URL: https://www.postgresql.org/docs/current/planner-optimizer.html (дата звернення: 18.03.2025).

5. Oracle database SQL tuning guide. *Oracle Corporation*. URL: https://docs.oracle.com/en/database/ (дата звернення: 18.03.2025).