УДК 004.658.2

Ігор Чихіра к.т.н, доцент, Віталій Левицький, к.т.н., Артур Микитишин
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ЕТАПИ ОПТИМІЗАЦІЇ БАЗ ДАНИХ

У роботі розглянуто аспекти оптимізації баз даних, що дозволяє прискорити операції вибірки та модифікації даними.

Ключові слова: база даних, запити, індекси, оптимізація

Igor Chyhira, Vitaliy Levytskyy, Artur Mykytyshyn

STAGES OPTIMIZATION BASED DATA

Abstract. The paper considers the aspects of database optimization, which allows to speed up the sampling and modification of data.

Keywords: database, queries, indices, optimization.

Характерними особливостями баз даних (БД) є:
- незалежність даних від програм;
- для даних допускається така мінімальна надлишковість, яка сприяє їх оптимальному використанню в одному чи кількох застосуваннях;
- для пошуку та модифікації даних використовуються спільні механізми;
- як правило, у складі БД існують засоби для підтримки її цілісності та захисту від неавторизованого доступу.

Оптимізація баз даних необхідна, коли в БД постійно додаються і видаляються записи. Таким чином оптимізація таблиць БД дозволяє прибрати «порожні» ключі, тим самим прискорюючи в майбутньому операції вибірки.

Можливі такі методи оптимізації БД:
- безпосередньо сама оптимізація БД і СУБД в цілому;
- оптимізація взаємодії програми та MS SQL Server;
- оптимізація запитів.

При оптимізації БД в цілому необхідно уникати: застою процедурного кешу, неоптимальних індексів та статистики. Виникає потреба у регламентних роботах в час мінімальної загрузки або проводити відповідні процедури на резервному сервері. Результативною буде робота при початковій оптимізації індексів і в подальшому очищенню процедурного кешу та оновлення статистики. Процес очистки процедурного кешу проводять командою: dbcc flushprocind ('ім'я бази даних'). Дана очистка корисна для БД з об'ємом до 500 Gb та при частому оновленню даних у базі даних. Оптимізація індексів включає фактори, які потрібно уникнути: сильно фрагментовані індекси, невикористовуванні індекси, відсутні індекси, індекси, які для свого обслуговування використовують більше витрат, ніж приносять виграш в продуктивності. Слід відзначити те, що навіть якщо індекс не використовується або вимагає значних витрат, то не потрібно поспішати його видаляти. Необхідно переконатися в тому, що він дійсно не потрібен системі. Для цього необхідно на тестовому середовищі, яке максимально наближене до виробничого, провести необхідні випробування-спочатку з індексом, а потім з віддаленим індексом. Таким чином для автоматизації процесу оптимізації індексів можна зробити наступні кроки: для кожної потрібної БД визначити

представлення і збережену процедуру, створити завдання в агента на періодичний виклик збереженої процедури з БД по адмініструванню. Важливо не забувати оновлювати статистику після реорганізації індексу, так як в такому випадку вона не оновлюється, а також при виконанні очищенню процедурного кешу [1].

При методі оптимізації взаємодії програми та MS SQL Server потрібно:
- працювати не з рядком, а з набором при відправці команд в БД; - асинхронно відправляти запити до БД і не змушувати користувача чекати відгуку програми;
- відправляти запити до БД комплексно, а не поодиноко (особливо актуально при зміні даних); -проводити кешування для всіх компонентів программного забезпечення, щоб зменшити звернення до БД; - правильно задавати рівень ізоляції при роботі з БД; - на серверній стороні програмного забезпечення при необхідності реалізовувати чергу звернень до БД; - не проводити фільтрацію по великого масиву даних не на стороні системи управління БД.

Наступним етапом оптимізації баз даних є вимоги до покращення створення запитів. Загальними рекомендаціями щодо оптимізації самих запитів є:
- заздалегідь якомога оптимальне фільтрування даних ще до моменту з'єднання їх з іншими таблицями;
- якомога менше за обсягом даних сортувати у результуючий набір;
- по можливості уникати конструкції DISTINCT, LIKE '% ...', OUTER JOIN особливо на великих об'ємах даних;
- якщо у вибірці потрібно лише одне поле від приєднаної таблиці, то не потрібно приєднувати таблицю, а в самій вибірці зробити підзапит;
- при фільтрації, агрегації і вибірці потрібно враховувати наявні індекси, щоб оптимізатор міг ними скористатися;
- повертати тільки ті поля, які дійсно потрібні, а не всі поля з усіх з'єднаних таблиць;
- не перевантажувати умови для з'єднання таблиць, а краще винести частину умови в фільтр [2].

Слід зауважити, що при зростанні розмірів інформаційної системи, зокрема кількості одночасних користувачів і обсягів самої БД, варто продумати поділ систем на OLTP і OLAP, де фонові завдання будуть виконуватися з OLAP-системою, а завдання реального часу, які виходять від користувачів, будуть виконуватися з OLTP-системою. Такий поділ дозволить ще більш чітко налаштувати кожну з систем і в значній мірі знизить навантаження на OLTP-систему [3].

Використання даних методів дозволить оптимізувати роботу системи управління базою даних з врахуванням подальшого розвитку інформаційної системи та збільшенням кількості одночасних звернень до БД.

Література.

1. Джеймс Р. Грофф, Пол Н. Вайнберг. SQL: полное руководство: пер. с англ. – К.: Издательская группа BHV, 2003. – 608с.
2. Ковязин А., Востриков С. Мир InterBase. – М.: Кудиц-Образ, 2004. – 560с.
3. Артеменко Ю.Н. MySQL. Руководство администратора — М.: Вильямс, 2005. — 624 с.