УДК 004.652.4 DOI: 10.30748/zhups.2018.57.19

С.В. Мінухін, В.Ю. В'юненко

Харківський національний економічний університет ім. Семена Кузнеця, Харків

ДОСЛІДЖЕННЯ ВПЛИВУ ВИКОРИСТАННЯ ІНДЕКСІВ В ТАБЛИЦЯХ РЕЛЯЦІЙНИХ БАЗ ДАНИХ

Досліджено вплив структури таблиць бази даних Microsoft SQL Server на швидкість виконання запитів до цих таблиць, а також вплив використання індексів на швидкість виконання запитів вибірки даних. За результатами дослідження визначено рекомендації щодо вибору структури таблиць баз даних та використання кластеризованих та некластеризованих індексів для підвищення продуктивності виконання запитів додавання, видалення та вибірки даних. Результати дослідження можуть бути використані для подальшої розробки бази даних з використанням індексів для отримання гарних показників продуктивності виконання запитів.

Ключові слова: *база даних, індексація, кластеризований індекс, некластеризований індекс, кластеризована таблиця, купа, продуктивність запитів.*

Вступ

Постановка проблеми. Актуальність даної теми пов'язана зі стрімким розвитком різноманітних систем керування базами даних (СКБД) у зв'язку з посиленням вимог до задач, які вирішуються за їх допомогою. Особливістю сучасних інформаційно-комунікаційних систем є велике навантаження на бази даних та великі розміри баз даних, через що виникають проблеми недостатньої продуктивності при виконанні запитів. Ці проблеми є актуальними для популярних хмарних платформ Microsoft Azure та Amazon Web Services при роботі з хмарними базами даних, також для соціальних мереж, великих торгових інтернет-майданчиків, баз даних онлайн-ігор, додатків для відстеження поштових відправлень, банківських додатків. Зокрема можна виділити наступні соціальні мережі, які в якості сховищ даних використовують реляційні СКБД: PostgreSQL – в Instagram, MySQL – в Facebook та Twitter. У зв'язку з постійним збільшенням кількості користувачів соціальних мереж та зростанням їх активності збільшується розмір баз даних та навантаження на них. В 2011 році, коли кількість користувачів Facebook становила 800 мільйонів, навантаження на бази даних – 60 мільйонів запитів за секунду. В 2018 році кількість користувачів Facebook зросла до 2,13 мільярдів, а Instagram до 1,1 мільярду, що посилює проблему зниження продуктивності виконання запитів.

За даними DB-Engines Ranking [1] лідируючі місця в списку найбільш популярних СКБД займають Oracle, MySQL та Microsoft SQL Server, які є реляційними. Тому для дослідження обрана СКБД Microsoft SQL Server.

Аналіз останніх досліджень і публікацій. У зв'язку з описаною проблемою необхідності підвищення продуктивності СКБД протягом останніх років проводиться багато досліджень. Після дослідження ефективної обробки запитів на великих просторових базах даних було виявлено, що індекси прискорюють час виконання запитів введення та виведення даних [2]. В наукових роботах визначаються нові напрями налаштування продуктивності бази даних, існуючі методи та підходи до оптимізації продуктивності БД та доводиться, що такі підходи мають бути переглянуті для підтримки запитів, що розроблені за допомогою інструментів об'єктно-реляційного відображення (ORM) [3]. Запропонована нова технологія індексації розподілених баз даних на основі двигуна індексації Lucene та доведено, що ця технологія надає прозорість масштабування розподіленої архітектури та зовнішніх пошукових індексів, оптимізує дані протоколу обміну між вузлами [4]. В роботі [5] пропонується ефективна технологія управління індексами для бази даних в реальному часі. У роботі [6] створення зручних та ефективних індексів в значній мірі сприяє підвищенню продуктивності запитів. Для зменшення витрат часу на введення-виведення даних можна використовувати кешування, нарощувати обробні потужності та встановлювати швидкодіючі жорсткі диски. У роботах [7–9] доведено, що створення індексів при правильному їх проектуванні може підвищити продуктивність системи обробки даних, для забезпечення ефективного доступу до даних.

На швидкість виконання стандартних операцій бази даних впливає структура її таблиць. Якщо первинний ключ таблиці створений на основі кластеризованого індексу, то така таблиця є кластеризованою, тобто зберігає свої сторінки з даними у вигляді списку та впорядковує їх у відповідності з послідовністю її індексної структури [10]. Якщо таблиця не має кластеризованого індексу, то її називають купою. Дані зберігаються в купі без вказування по-

рядку [11]. В багатьох літературних джерелах по сучасним методам роботи з SQL Server, стверджується, що кожна таблиця повинна мати кластеризований індекс і не бути купою [12]. Таким чином, одним із способів підвищення продуктивності запитів є створення індексів.

Мета статті – проаналізувати вплив вибору структури таблиць баз даних Microsoft SQL Server та використання кластеризованих та некластеризованих індексів на продуктивність виконання запитів додавання, видалення та вибірки даних.

Для досягнення поставленої мети пропонується вирішити такі задачі:

1. Дослідити вплив структури таблиць бази даних на швидкість виконання запитів до цих таблиць.

2. Дослідити вплив використання індексів на швидкість виконання запитів вибірки даних.

3. Провести аналіз отриманих результатів.

Виклад основного матеріалу

Індекси створюються для стовпців таблиць та надають шлях для швидкого пошуку даних на основі значень в цих стовпцях. Наприклад, якщо буде створений індекс по первинному ключу, а потім буде виконаний пошук рядка з даними, використовуючи значення первинного ключа, то SQL Server спочатку знайде значення індексу, а потім використає індекс для швидкого знаходження всього рядка з даними [13]. Без індексу буде виконаний повний перегляд (сканування) всіх рядків таблиці, що може значно впливати на продуктивність.

Індекси компонента Database Engine створюються, використовуючи структуру B-дерева (збалансованого дерева). B-дерево має деревовидну структуру, в якій всі найнижчі вузли (листя) знаходяться на відстані однакової кількості рівнів від вершини (кореневого вузла) дерева. Ця властивість підтримується і тоді, коли в індексований стовпчик додаються чи видаляються дані [9].

При формуванні запиту на індексований стовпчик підсистема запитів починає йти зверху від кореневого вузла вниз через проміжні вузли, при цьому кожен шар проміжного рівня містить більш детальну інформацію про дані [14]. Підсистема запитів продовжує рухатися по вузлах індексу до тих пір, поки не досягне нижнього рівня з листям індексу. Листя індексу можуть містити як самі дані таблиці, так і просто покажчик на рядки з даними в таблиці, в залежності від типу індексу: кластеризований індекс або некластеризований [15].

Кластеризовані індекси сортують і зберігають рядки даних в таблицях або представленнях на основі їх ключових значень. Кластеризований індекс створюється за замовчуванням для кожної таблиці, для якої визначений первинний ключ [9]. Така таблиця називається кластеризованою. Таблиця, для якої не визначений первинний ключ, називається купою.

Некластеризований індекс – це окрема структура, як правило, виду B-дерева, яка створюється додатково до основної таблиці [16]. Некластеризовані індекси мають окрему від рядків даних структуру. У некластеризованому індексі містяться значення ключа некластеризованого індексу, і кожен запис значення ключа містить покажчик на рядок даних, що містить значення ключа [17].

Пошук даних можна здійснювати двома способами в залежності від типу таблиці:

1. Купа – проходження при пошуку по структурі некластеризованого індексу, після чого рядок витягується, використовуючи ідентифікатор рядка.

2. Кластеризована таблиця – проходження по структурі некластеризованого індексу, після чого відбувається проходження по відповідному кластеризованому індексу.

Для дослідження продуктивності при додаванні даних в таблицю з кластеризованим індексом і без (до кластеризованої таблиці та купи) було створено базу даних з двома тестовими таблицями, одна з яких має первинний ключ на основі кластеризованого індексу, а в другій первинний ключ створений за допомогою некластеризованого індексу (таблиця представляє собою купу). Таблиці були заповнені однаковими тестовими даними по 5 млн рядків.

В результаті для цієї бази даних було зарезервовано на диску 3,3 Гб. Розмір даних в обох таблицях однаковий – по 251 Мб, а розміри індексів становлять: кластеризований – 0,9 Мб, некластеризований – 87,3 Мб.

В процесі дослідження виконувалися такі операції виконання запитів:

1. Операція 1 – додавання по 100 000 записів до таблиць.

2. Операція 2 – додавання по 100 000 записів до таблиць після попереднього видалення по 500 000 записів.

3. Операція 3 – додавання по 100 000 записів до таблиць.

4. Операція 4 – додавання по 100 000 записів до таблиць після попереднього видалення по 1 000 000 записів.

5. Операція 5 – додавання по 200 000 записів до таблиць.

Дослідження виконувалося на локальному ресурсі з такими характеристиками: процесор Intel Core i3-2370M CPU 2.40 GHz, RAM 6 Гб. Для створення та управління базами даних використовувався Microsoft Server 2016 та середовище SQL Server Management Studio, а також програма SQLQueryStress [18] для аналізу часу виконання запитів.

Монітор активності в середовищі SQL Server Management Studio дозволяє створювати динамічні представлення поточної активності. Результати моніторингу активності під час виконання запитів для виконання операцій 1–5 представлені в табл. 1, в якій ЦП – рівень завантаження ЦП запитом; операцій читання – швидкість логічних операцій читання для запиту; операцій запису – швидкість логічних операцій запису для запиту.

Як видно з табл. 1, майже у всіх випадках показники, що відповідають кластеризованій таблиці, є кращими. Це пов'язано з тим, що при вставці даних в купу потрібне оновлення двох об'єктів – некластеризованого індексу і самої таблиці, що вимагає додаткових ресурсів від ЦП, та в цей момент виконується більше операцій читання і запису, ніж при вставленні даних в таблицю з кластеризованим індексом, отже такий запит виконується довше. Значний розрив показників є у тих операціях, коли додавалися записи після попереднього видалення. Ця різниця виникла через те, що при вставці записів в купу СКБД шукає порожній простір на кожній сторінці для розміщення в ньому даних [19]. Причиною цього є той факт, що дані купи не відсортовані, отже записи можна розміщувати куди завгодно. В таблиці з кластеризованим індексом при нарощуванні значення первинного ключа вставка завжди здійснюється в кінець таблиці. В результаті обсяг даних кластеризованої таблиці збільшується, а обсяг даних у купі не змінюється.

Таблиця 1

Показники монітору активності при виконанні запитів

Номер операції	Тип таблиці	ЦП (мс/с)	Операцій читання	Операцій запису	Тривалість (мс)
1	Кластеризована	10344	360142	601	64437
	Купа	13117	584850	966	71041
	Відношення	1,2681	1,6239	1,6073	1,1025
2	Кластеризована	19159	404226	99029	88069
	Купа	52027	606438	92497	98537
	Відношення	2,7155	1,5002	0,9340	1,1189
3	Кластеризована	13381	367735	657	75721
	Купа	37359	657336	12367	82338
	Відношення	2,7919	1,7875	18,8234	1,0874
4	Кластеризована	11372	360666	4575	66655
	Купа	13337	585525	5992	70968
	Відношення	1,1728	1,6235	1,3097	1,0647
5	Кластеризована	1565	69186	718	8067
	Купа	28930	1240900	10802	148103
	Відношення	18,4856	17,9357	15,0446	18,3591

Рядок «Відношення» в табл. 1 показує відношення отриманих результатів, що стосуються купи, до результатів, щодо кластеризованої таблиці. При виконанні деяких запитів показники, що стосуються кластеризованої таблиці, менші приблизно у 18 разів.

За результатами дослідження побудовані графіки, які відображають показники моніторингу активності при виконанні запитів запису даних в кластеризовану таблицю та в купу. На графіках відображені залежності рівня завантаження ЦП запитами запису даних (рис. 1), швидкості логічних операцій читання для запитів запису даних (рис. 2), швидкості логічних операцій запису для запитів запису даних (рис. 3) та тривалість виконання запитів запису даних (рис. 4) від типу операції, що виконується.

Проаналізувавши графіки, можна зробити висновки, що рівень завантаження ЦП запитами запису даних значно більший для купи; швидкість логічних операцій читання для запитів запису даних менша для кластеризованої таблиці; швидкість логічних операцій запису менша для кластеризованої таблиці; тривалість виконання запитів запису даних менша для кластеризованої таблиці. Отже, продуктивність виконання запитів до кластеризованої таблиці більша у 18 разів (для операції 5).

Рис. 1. Залежність рівня завантаження ЦП запитами запису даних від типу операції, що виконується

Рис. 2. Залежність швидкості логічних операцій читання для запитів запису даних від типу операції, що виконується

Рис. 3. Залежність швидкості логічних операцій запису для запитів запису даних від типу операції, що виконується

Рис. 4. Залежність тривалості виконання запитів запису даних від типу операції, що виконується

В табл. 2–3 наведені результати тестування продуктивності запитів вибірки даних з кластеризованої таблиці та купи з використанням програми SQLQueryStress. Рядок «Відношення» в табл. 2–3 показує відношення отриманих результатів, що стосуються купи, до результатів, щодо кластеризованої таблиці.

Виходячи з результатів можна сказати, що запити вибірки даних для кластеризованої таблиці виконуються швидше, а також повторні запити з тими ж параметрами збільшують швидкість пошуку.

Таблиця 2

Показники SQLQueryStress при виконанні запиту для пошуку рядка по первинному ключу, вказуючи на конкретний номер рядка

Тип таблиці	Кількість ітерацій/кількість потоків					
	1/1	1/1	10/5	10/5	20/20	20/20
Кластеризована	0,0269	0,0258	0,3620	0,3559	2,5582	2,3753
Купа	0,1366	0,1118	0,3667	0,3618	2,8367	2,5929
Відношення	5,0781	4,3333	1,0130	1,0166	1,1089	1,0916

Таблиця 3

Показники SQLQueryStress при виконанні запиту для пошуку рядків по первинному ключу, вказуючи на діапазон рядків

Тип таблиці	Кількість ітерацій/кількість потоків					
	1/1	1/1	10/5	10/5	20/20	20/20
Кластеризована	0,5222	0,1838	6,3901	6,3085	50,7996	49,2985
Купа	1,2408	0,4948	15,2905	17,1294	112,2537	109,9601
Відношення	2,3761	2,6921	2,3928	2,7153	2,2097	2,2305

Для дослідження впливу використання індексів для таблиць бази даних на продуктивність виконання запитів вибірки були розроблені дві бази даних, які є моделями типових баз даних для торгівельних підприємств, з однаковою структурою та різним розміром (9,7 Гб та 876 Мб). В ході дослідження було виконано пошук рядка по первинному ключу, вказуючи на конкретний номер рядка, та пошук рядків по первинному ключу, вказуючи на діапазон значень первинного ключа. В результаті для повторних запитів з однаковими параметрами швидкість пошуку кількісно зменшувалася в діапазоні від 6,7 разів до 1,2 разу. Запити для таблиці бази даних великого розміру виконувалися в 1,1 раз довше ніж для таблиці бази даних меншого розміру.

Далі було виконано пошук рядків по шаблону з індексованими і непроіндексованими стовпцями, а також пошук рядків по даті з індексованими і непро-

індексованими стовпцями. Після проведення дослідження було виявлено, що запити вибірки даних з використанням індексів виконуються швидше. Після створення індексів до стовпчиків таблиці бази даних час виконання запиту пошуку рядків по шаблону зменшився в 1,5 рази, а час виконання запиту пошуку рядків по даті зменшився в 2,5 рази.

Таким чином можна сформулювати такі рекомендації щодо планування стратегії індексування:

1. Для таблиць які часто оновлюються необхідно використовувати якомога менше індексів.

2. Якщо таблиця містить велику кількість даних, які не змінюються, тоді можна використовувати стільки індексів, скільки необхідно для поліпшення продуктивності запитів. Однак потрібно ретельно зважити переваги перед використанням індексів на невеликих таблицях, тому що використання пошуку за індексом може зайняти більше часу, ніж просте сканування всіх рядків.

3. Унікальність значень в стовпчику впливає на продуктивність індексу. У загальному випадку, чим більше дублікатів в стовпчику, тим гірше працює індекс. З іншого боку, чим більше унікальних значення, тим вище "працездатність" індексу.

4. Для оптимальної продуктивності запитів індекси зазвичай створюються на тих стовпцях таблиці, які часто використовуються в запитах. Для однієї таблиці може бути створено кілька індексів. Однак збільшення числа індексів уповільнює операції додавання, оновлення, видалення рядків таблиці, оскільки при цьому доводиться оновлювати самі індекси. Крім того, індекси займають додатковий обсяг пам'яті, тому перед створенням індексу слід переконатися, що потенційний виграш в продуктивності запитів перевищить додаткову витрату ресурсів на супровід індексу.

Висновки

За результатами проведеного дослідження відповідно до зазначених задач можна визначити наступне:

1. Для підвищення продуктивності виконання запитів видалення, додавання і вибірки до баз даних великих об'ємів потрібно зберігати дані у вигляді кластеризованої таблиці. При виконанні деяких запитів показники завантаженості ЦП, кількості операцій читання та запису даних та тривалості виконання запитів, що отримані для кластеризованої таблиці, менші у 18 разів.

2. Вищу продуктивність мають запити з використанням проіндексованих стовпців. Час виконання запиту пошуку рядків по шаблону зменшився в 1,5 рази, а час виконання запиту пошуку рядків по даті – в 2,5 рази.

Для підвищення продуктивності виконання запитів додавання, видалення та вибірки даних пропонується: зберігати дані у вигляді купи для таблиць невеликих розмірів; зберігати дані у вигляді кластеризованої таблиці для таблиць з великою кількістю даних, коли необхідно виконувати запити частої вибірки діапазонів даних; для таблиць які часто оновлюються треба створювати велику кількість індексів; створювати індекси для тих стовпців таблиці, які часто використовуються в запитах.

Результати дослідження можуть бути використані для подальшої розробки бази даних з використанням індексів для підвищення продуктивності виконання запитів в розподілених системах та хмарних обчисленнях.

Список літератури

1. DB-Engines Ranking [Електронний ресурс]. – Режим доступу до ресурсу: https://db-engines.com/ en/ranking.

2. Roumelis G. Efficient query processing on large spatial databases: A performance study / G. Roumelis, M. Vassilakopoulos, A. Corral, Y. Manolopoulos // Journal of Systems and Software. – 2017. – Vol. 132. – P. 165-185. https://doi.org/10.1016/j.jss.2017.07.005

3. Colley D. Identifying New Directions in Database Performance Tuning / D. Colley, Dr. Clare Stanier // Procedia Computer Science. – 2017. – Vol. 121. – P. 260-265. https://doi.org/10.1016/j.procs.2017.11.036

4. Hassen F. An efficient synchronous indexing technique for full-text retrieval in distributed databases / F. Hassen, Grissa Touzi Amel // Procedia Computer Science. – 2017. – Vol. 112. – P. 811-821. https://doi.org/10.1016/ j.procs.2017.08.071

5. Meng Long Yue A Real-Time Database Index Management Method for Fast Access to Massive Data of Power System / Meng Long Yue, Cui Hui Ren, Xian Hui Li // Procedia Engineering. – 2017. – Vol. 24. – P. 165-170. https://doi.org/10.1016/j.proeng.2011.11.2620.

6. Хендерсон К. Microsoft SQL Server: структура и реализация. Профессиональное руководство / К. Хендерсон. – СПб.: БХВ-Петербург, 2005. – 560 с.: ил.

7. Бондарь А.Г. Microsoft SQL Server 2014 / А.Г. Бондарь. – СПб.: БХВ-Петербург, 2015. – 592 с.: ил.

8. Мирошниченко Г.А. Реляционные базы данных: практические приемы оптимальных решений / Г.А. Мирошниченко. – СПб.: БХВ-Петербург, 2005. – 400 с.: ил.

9. Petkovic D. Microsoft SQL Server 2012. A beginner's guide / D. Petkovic. – The McGraw-Hill Companies, 2012. – 833 p.

10. Кластеризованные индексы в SQL Server: Вы должны это знать. [Електронний ресурс]. – Режим доступу до ресурсу: http://www.sqlbooks.ru/readarticle.aspx?part=01&file=design03.

11. Heaps (Tables without Clustered Indexes) [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-gb/sql/relational-databases/indexes/heaps-tables-without-clustered-indexes?view=sql-server-2017.

12. Ben Snaidero. SQL Server Insert Performance for Clustered Indexes vs. Heap Tables [Електронний ресурс] / Snaidero Ben. – Режим доступу до ресурсу: https://www.mssqltips.com/sqlservertip/4961/sql-server-insert-performance-for-clustered-indexes-vs-heap-tables/.

13. Clustered and Nonclustered Indexes Described [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-gb/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-2017.

14. Create Clustered Indexes [Електронний ресурс]. – Режим доступу до ресурсу: https://docs.microsoft.com/en-gb/sql/relational-databases/indexes/create-clustered-indexes?view=sql-server-2017.

15. Sheldon Robert 14 SQL Server Indexing Questions You Were Too Shy To Ask / Robert Sheldon. – [Електронний ресурс]. – Режим доступу до ресурсу: https://www.red-gate.com/simple-talk/sql/performance/14-sql-server-indexing-questions-you-were-too-shy-to-ask/.

16. Для чего НЕ нужны индексы [Електронний ресурс]. – Режим доступу до ресурсу: https://infostart.ru/public/444987/.

17. Повышение производительности запроса [Електронний ресурс]. – Режим доступу до ресурсу: https://www.intuit.ru/studies/courses/1079/264/lecture/6721.

18. SQL query stress simulator created by Adam Machanic [Електронний ресурс]. – Режим доступу до ресурсу: https://github.com/ErikEJ/SqlQueryStress.

19. Повышение производительности баз данных: Практические советы [Електронний ресурс]. – Режим доступу до ресурсу: https://www.ixbt.com/live/diy/povyshenie-proizvoditelnosti-baz-dannyh-prakticheskie-sovety.html.

References

1. *DB-Engines Ranking*, https://db-engines.com/en/ranking (accessed 2 July 2018).

2. Roumelis, G., Vassilakopoulos, M., Corral, A. and Manolopoulos, Y. (2017), Efficient query processing on large spatial databases: A performance study, *Journal of Systems and Software*, Vol. 132, pp. 165-185. https://doi.org/10.1016/j.jss.2017.07.005.

3. Colley, D. and Dr. Clare Stanier (2017), Identifying New Directions in Database Performance Tuning, *Procedia Computer Science*, Vol. 121, pp. 260-265. https://doi.org/10.1016/j.procs.2017.11.036.

4. Hassen, F. and Grissa Touzi Amel (2017), An efficient synchronous indexing technique for full-text retrieval in distributed databases, *Procedia Computer Science*, Vol. 112, pp. 811-821. https://doi.org/10.1016/ j.procs.2017.08.071.

5. Meng Long Yue, Cui Hui Ren and Xian Hui Li (2017), A Real-Time Database Index Management Method for Fast Access to Massive Data of Power System, *Procedia Engineering*, Vol. 24, pp. 165-170. https://doi.org/10.1016/j.proeng.2011.11.2620.

6. Khenderson, K. (2005), "*Microsoft SQL Server: struktura y realyzacyja. Professyonaljnoe rukovodstvo*" [*Microsoft SQL Server: structure and implementation. Professional guidance*], BKhV-Peterburgh, St. Petersburg, 560 p.

7. Bondarj, A.Gh. (2015), *Microsoft SQL Server 2014*, BKhV-Peterburgh, St. Petersburg, 592 p.

8. Myroshnychenko, Gh.A. (2005), "*Reljacyonnye bazy dannykh: praktycheskye pryemy optymaljnykh reshenyj*" [*Relational databases: practical methods of optimal solutions*], BKhV-Peterburgh, St. Petersburg, 400 p.

9. Petkovic, D. (2012), *Microsoft SQL Server 2012. A beginner's guide*, The McGraw-Hill Companies, 833 p.

10. "Klasteryzovannye Yndeksy v SQL Server: Vy dolzhny eto znatj" [*Clustered Indexes in SQL Server: You need to know*], http://www.sqlbooks.ru/readarticle.aspx?part=01&file=design03 (accessed 27 June 2018).

11. *Heaps (Tables without Clustered Indexes)*, https://docs.microsoft.com/en-gb/sql/relational-databases/indexes/heaps-tables-without-clustered-indexes?view=sql-server-2017 (accessed 27 June 2018).

12. Ben Snaidero. *SQL Server Insert Performance for Clustered Indexes vs. Heap Tables*, https://www.mssqltips.com/sqlservertip/4961/sql-server-insert-performance-for-clustered-indexes-vs-heap-tables/ (accessed 29 June 2018).

13. *Clustered and Nonclustered Indexes Described*, https://docs.microsoft.com/en-gb/sql/relational-databases/indexes/clustered-and-nonclustered-indexes-described?view=sql-server-2017 (accessed 30 June 2018).

14. *Create Clustered Indexes*, https://docs.microsoft.com/en-gb/sql/relational-databases/indexes/create-clustered-indexes?view=sql-server-2017 (accessed 2 July 2018).

15. Sheldon, Robert (2018), *14 SQL Server Indexing Questions You Were Too Shy To Ask*, https://www.red-gate.com/simple-talk/sql/performance/14-sql-server-indexing-questions-you-were-too-shy-to-ask/ (accessed 2 July 2018).

16. (2018), "Dlja chegho NE nuzhny yndeksy" [*Why do NOT need indexes*], https://infostart.ru/public/444987/ (accessed 3 July 2018).

17. (2018), "Povyshenye proyzvodyteljnosty zaprosa" [*Increase query performance*], https://www.intuit.ru/studies/courses/1079/264/lecture/6721 (accessed 5 July 2018).

18. (2018), *SQL query stress simulator created by Adam Machanic*, https://github.com/ErikEJ/SqlQueryStress (accessed 5 July 2018).

19. (2018), "Povyshenye proyzvodyteljnosty baz dannykh: Praktycheskye sovety" [*Improving database performance: Practical tips*], https://www.ixbt.com/live/diy/povyshenie-proizvoditelnosti-baz-dannyh-prakticheskie-sovety.html (accessed 5 July 2018).

Надійшла до редколегії 19.06.2018
Схвалена до друку 17.07.2018

Відомості про авторів:

Мінухін Сергій Володимирович
доктор технічних наук
професор
Харківського національного економічного
університету ім. Семена Кузнеця,
Харків, Україна
https://orcid.org/0000-0002-9314-3750

В'юненко Вікторія Юріївна
магістрант
Харківського національного економічного
університету ім. Семена Кузнеця,
Харків, Україна
https://orcid.org/0000-0001-8807-7317

Information about the authors:

Sergii Minukhin
Doctor of Technical Sciences
Professor
of Simon Kuznets Kharkiv
National University of Economics,
Kharkiv, Ukraine
https://orcid.org/0000-0002-9314-3750

Viktoriia Viunenko
Graduate Student
of Simon Kuznets Kharkiv
National University of Economics,
Kharkiv, Ukraine
https://orcid.org/0000-0001-8807-7317

ИССЛЕДОВАНИЕ ВЛИЯНИЯ ИСПОЛЬЗОВАНИЯ ИНДЕКСОВ В ТАБЛИЦАХ РЕЛЯЦИОННЫХ БАЗ ДАННЫХ

С.В. Минухин, В.Ю. Вьюненко

Исследовано влияние структуры таблиц базы данных Microsoft SQL Server на скорость выполнения запросов к этим таблицам, а также влияние использования индексов на скорость выполнения запросов выборки данных. По результатам исследования определены рекомендации по выбору структуры таблиц баз данных и использования кластеризованных и некластеризованных индексов для повышения производительности выполнения запросов добавления, удаления и выборки данных. Результаты исследования могут быть использованы для дальнейшей разработки базы данных с использованием индексов для получения хороших показателей производительности выполнения запросов.

Ключевые слова: *база данных, индексация, кластеризованный индекс, некластеризованный индекс, кластеризованная таблица, куча, производительность запросов.*

INVESTIGATION OF THE INFLUENCE OF USE OF INDEXES IN TABLES RELATIONAL DATABASES

S. Minukhin, V. Viunenko

Influence of structure of tables of the database of Microsoft SQL Server on the speed of execution of requests to these tables and also influence of use of indexes on the speed of execution of requests of data selecting is probed. The research was executed locally. For creation and control of databases Microsoft Server 2016 and the environment SQL Server Management Studio to manage SQL Server and also the SQLQueryStress program for the analysis of execution of requests was used. For check of productivity when adding data in the table with clustered by the index and without (in the clustered table and a heap) the database with two test tables, one of which has primary key on the basis of the clustered index, was created, and in the second primary key it is created by means of non-clustered index and the table represents a heap. The research included execution of delete queries and adding of records in tables and also execution of requests of data selecting. The received results are presented in the form of the table in which indices of monitoring of activity. Also for evident display of results diagrams are constructed. Two databases which are models of typical databases for trade enterprises, with identical structure and the different size were developed for a research of influence of use of indexes for tables of the database on productivity of execution of requests of selection. By results of a research recommendations for increase in productivity of execution of requests of adding, deleting and data selecting are defined. Results of a research can be used for further development of the database with use of indexes for receiving good performance measures of execution of requests.

Keywords: *database, indexing, clustered index, non-clustered index, clustered table, heap, query performance.*