УДК 004.457/004.054

ПРОГРАММНОЕ СРЕДСТВО ИССЛЕДОВАНИЯ РАЗЛИЧНЫХ АЛГОРИТМОВ КЭШИРОВАНИЯ В РЕЛЯЦИОННЫХ СУБД

Иванов В. И., Новиков С. П.
Донской государственный технический университет, г. Ростов-на-Дону, Российская Федерация
slava-ivanoff@mail.ru,
n_serg7@mail.ru.

Рассмотрены различные алгоритмы кэширования. Реализовано приложение, позволяющее определить эффективность того или иного метода в конкретно заданной реляционной базе данных путём вывода затраченного времени на SQL-запросы.

Ключевые слова: кэш, кэширование, реляционная база данных.

UDC 004.457/004.054

SOFTWARE TOOL FOR INVESTIGATING VARIOUS CACHING ALGORITHMS IN RELATIONAL DBMS

Ivanov V. I., Novikov S. P.
Don State Technical University, Rostov-on-Don, Russian Federation
slava-ivanoff@mail.ru,
n_serg7@mail.ru.

The article examines various caching algorithms. The authors have implemented an application, which allows them to determine the effectiveness of a particular method in a specified relational database, by outputting the time spent on SQL queries.

Keywords: cache, caching, relational database

Введение. В настоящее время все больше сфер деятельности использует базы данных (БД). Их развитие идет стремительными темпами, а объемы хранимой информации пропорционально возрастают. Благодаря исследованиям частных компаний было выяснено, что наибольшую популярность в данный момент имеют реляционные базы данных — MySQL, PostgreSQL, MS SQL server. Реляционная база данных — это совокупность взаимосвязанных таблиц, каждая из которых содержит информацию об объектах определенного типа. Строка таблицы содержит данные об одном объекте, а столбцы таблицы описывают различные характеристики этих объектов — атрибутов. Записи, то есть строки таблицы, имеют одинаковую структуру — они состоят из полей, хранящих атрибуты объекта. Каждое поле, то есть столбец, описывает только одну характеристику объекта и имеет строго определенный тип данных. Все записи имеют одни и те же поля, только в них отображаются различные информационные свойства объекта [1, 2, 3].

Одна из проблем, возникающих при работе с реляционными базами данных, — это потеря производительности из-за постоянно возрастающего объёма хранимых данных. Одним из способов решения данной проблемы является использование кэша и операции кэширования. Кэш (cache) — промежуточный буфер с быстрым доступом, содержащий информацию, которая может быть запрошена с наибольшей вероятностью. Соответственно кэширование — это процесс размещения и хранения какой-либо информации в кэше с целью увеличения скорости доступа к ней [2, 3]. Доступ к данным в кэше осуществляется быстрее, чем выборка исходных данных из более медленной памяти или удаленного источника, однако её объём существенно ограничен по сравнению с хранилищем исходных данных. Однако неправильно выбранный способ кэширования способен наоборот понизить производительность. Таким образом, необходимо средство, которое способно определять, какой именно алгоритм кэширования нужно применять в каждой отдельно взятой базе данных.

Постановка задачи. Необходимо рассмотреть основные используемые алгоритмы кэширования, разработать и реализовать программное средство, с помощью которого можно проводить тестирование различных алгоритмов кэширования в реляционной базе данных при

помощи SQL-запросов. При этом требуется использовать существующую систему управления базами данных (СУБД) и сохранить все базовые отношения в БД.

Описание метода. В данный момент основными используемыми алгоритмами кэширования являются: FIFO, LFU, LRU, сегментный LRU, mid point LRU, 2Q, MQ. Рассмотрим каждый из них по отдельности [1, 2, 3].

FIFO (first input first output) работает как обычная очередь по принципу «первый вошел, первый вышел». Т.е. если искомый элемент не находится в кэше, он вставляется в хвост очереди. Для освобождения места удаляются элементы из головы очереди.

В LFU для каждого элемента кэша создается счётчик обращений. При этом каждый новый элемент вставляется в кэш со значением счётчика равным 1 и значение его увеличивается с каждым новым обращением. Главным недостатком LFU является то, что некогда частотные элементы могут присутствовать в кэше очень долго, даже если уже давно не происходил их запрос.

При использовании LRU каждый новый элемент вставляется в голову списка. При попадании в кэш элемент перемещается в голову списка вытесняя оттуда старый элемент. Если нужно освободить место, вытесняется элемент из хвоста списка. Таким образом, вытесняется элемент, который не запрашивался дольше всех.

Сегментный LRU является модернизированной версией обычного LRU. Кэш организован в виде нескольких LRU кэшей. Новый элемент вставляется в нулевой LRU кэш. При попадании в кэш элемент перемещается в следующий LRU кэш, либо в голову списка последнего LRU кэша, если выше уже идти некуда. При вытеснении элемента из k-го LRU кэша он перемещается в k-1 LRU кэш. По достижению 0LRU кэша, элемент удаляется [3, 4].

Mid point LRU является подвидом сегментного LRU, в котором количество сегментов — 2, и эти два сегмента делят кэш не на пополам, а в пропорции, которая вычисляется для каждой базы данных опытным путём.

2Q — данный вид кэширования характерен тем, что разделен на 3 части: In, Out, Main. In — FIFO кэш, в который попадают все новые элементы. Out — FIFO кэш, в который попадают элементы, вытесненные из In. При этом этот кэш хранит ключ и не хранит значение, поэтому его можно сделать достаточно большим. Main — главный LRU кэш, в который попадают новые элементы, найденные в Out. При вытеснении элемента из главного кэша он удаляется [3, 4, 5].

MQ — является вариацией LRU кэша. Для каждого элемента ведётся счётчик его запросов. Номер LRU кэша, в который стоит поместить элемент, вычисляется используя некоторую функцию от этого счётчика, например, логарифм.

В силу особенностей кэш памяти (малого его количества) эффективнее всего применять операцию кэширования в случае, когда объем данных в таблицах не чрезмерно большой.

Программная реализация и результат. Для реализации данного метода и проведения сравнительных испытаний, была выбрана СУБД «PostgreSQL 9.6». В качестве примера была выбрана упрощенная локальная база данных банка, заполненная данными. Ее структура показана на рис. 1.

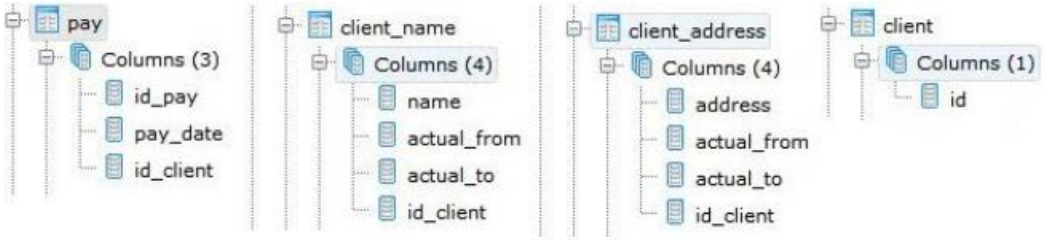

Рис. 1. Структура БД

Реализация приложения для тестирования различных видов кэширования осуществлена на языке программирования C#.

Все описанные в структуре программы алгоритмы тестировались про помощи автоматизированного составления 1000 запросов к структуре базы данных, состоящей из 10 000 записей. Это позволяет сказать о репрезентативности полученных результатов, которые отображены в таблице 1, а также на рис. 2. Данные отображены при расчете по среднему времени обработки запроса из 10 000 опытов и представлены в секундах.

Таблица 1

Результаты работы приложения по тестированию различных алгоритмов кэширования

Без кэша	FIFO	LFU	LRU	SLRU	mpLRU	2Q	MQ
10,361	10,765	9,972	10,293	10,453	10,454	10,785	10,532
10,41	10,687	10,001	10,285	10,345	10,351	10,738	10,583
10,334	10,794	10,031	10,215	10,495	10,49	10,741	10,594
10,359	10,698	9,983	10,243	10,462	10,462	10,79	10,523
10,318	10,715	10,114	10,363	10,501	10,516	10,816	10,578
10,361	10,536	10,056	10,339	10,434	10,428	10,804	10,573
10,395	10,621	10,04	10,298	10,483	10,489	10,809	10,604
10,297	10,695	10,155	10,257	10,429	10,428	10,764	10,542
10,297	10,748	9,99	10,343	10,504	10,513	10,753	10,537

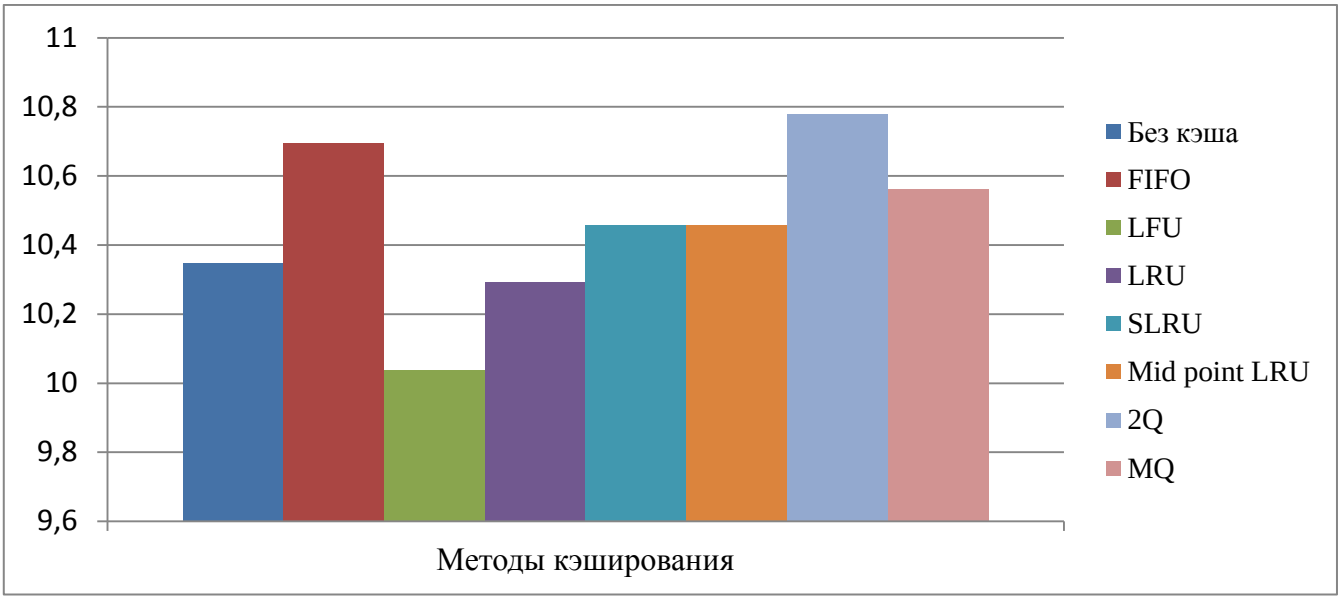

Рис. 2. График производительности алгоритмов кэширования

Как видно из графиков и таблицы, использование LFU и LRU в данном случае незначительно ускорили скорость выполнения SQL-запросов, а все остальные — замедлили. В качестве точки отсчёта для сравнения производительности при разных способах кэширования приходится значение «без кэша», т.е. это значение можно взять за 100%. В таком случае получаются значения, представленные в таблице 2.

Таблица 2

Сравнение полученных результатов

Без кэша	FIFO	LFU	LRU	SLRU	mpLRU	2Q	MQ
100%	103,33%	96,97%	99,44%	101,01%	101%	104,12%	102,04%

В данном случае при использовании LFU и LRU прирост производительности слишком мал и не позволяет сделать вывод о целесообразности использования кэша.

Заключение. Рассмотрены основные типы алгоритмов кэширования и создано приложение, с помощью которого можно тестировать эффективность использования того или иного алгоритма кэширования в реляционных базах данных. В данный момент в работе приложения остаются нерешенными проблемы осуществления выбора пользователем СУБД из нескольких заранее предложенных, с которой должна осуществляться работа, определения структуры базы данных для того чтобы в последующем можно было осуществлять более сложные и разнообразные SQL-запросы и остается проблема формирования не однотипных SQL-запросов.

Библиографический список

1. Реляционная база данных и её структура [электронный ресурс] / Научная библиотека избранных естественно-научных изданий. — Режим доступа: http://sernam.ru/book_cbd.php?id=2 (дата обращения 04.05.2017)

2. Федорова Г. Н. Разработка и администрирование баз данных / Г. Н. Федорова. — Москва: Академия, 2015. — 58с.

3. Дейт, К. Дж. Введение в системы баз данных, 8-е издание.: пер. с англ. / Дж. К. Дейт. — Москва: Издательский дом «Вильямс», 2005. — 1328 с.

4. Первое знакомство с кэшем и процессом кэширования [электронный ресурс] / Microsoft TechNet. — Режим доступа: https://social.technet.microsoft.com/wiki/ru-ru/contents/articles/11074.aspx (дата обращения: 04.05.2017).

5. Алгоритмы кэширования [электронный ресурс] / Программерский блог. — Режим доступа: https://artemy-kolesnikov.blogspot.ru/2014/11/blog-post.html (дата обращения 07.05.2017).