Отже запропонована методика керування акустичними параметрами без зміни архітектури приміщення та використання акустичної обробки матеріалами. Отримані результати дозволили визначити межі застосування керованої інтерференції звукових полів та відстань розміщення акустичних систем від стіни та між ними.

УДК 004.75

МЯКИШЕВ В.К., ДАНИЛЕНКО А.Ф., доц.

ТЕХНОЛОГИИ КЭШИРОВАНИЯ В WEB-ПРИЛОЖЕНИЯХ

Постановка проблемы. В последнее время веб-сайты становятся все более динамичными и интерактивными, требуют все больше ресурсов, а число их посетителей все растет, что приводит к осложнениям в плане производительности. Небольшая задержка, даже всего полсекунды после щелчка по ссылке – уже раздражает. Приходится закупать дополнительное оборудование, чтобы разгрузить старые машины, и вернуть мгновенное выполнение сценариев на сайтах. Сайты требуют и вычислительную мощность, и место на жестких дисках, однако пропорции развития сейчас таковы, что дешевле выходит дополнительная память, чем процессор. Поэтому распространено применение систем кэширования.

Однако разрабатывать для каждого проекта выгодную систему кэширования – трудоёмко, а приспосабливание уже существующих порой даже сложнее. Поэтому не смотря на очевидные выгоды и разработчиков (их продукт будет работать быстрее), и хостинговых компаний (меньше затрат на оборудование) этот пункт часто опускается в технических заданиях.

Одним из способов решений этой проблемы – системы автокэширования, которые есть в любой относительно серьезной Web-CMS [1]. Эти системы автоматически отслеживают адрес страницы и параметры, которые ей передаются, и запоминают вывод сценария при этих параметрах, и при возможности помещают его в кэш. Преимуществом является то, что кэширование будет работать с любой страницей, которая при одинаковых параметрах всегда выдает одинаковый результат, и в то же время программисты сценария этой страницы вовсе не задумываются о необходимости кэширования, а могут полностью сосредоточиться на основной задаче. Тем не менее, правильно спланированное кэширование для каждого отдельного случая всегда даст лучший результат.

Цель работы. Найти компромисс между затратой большого количества времени на разработку специальной системы кэширования, которые дают лучший вариант, и использованием полностью автоматических средств, которые дают лишь малый прирост производительности, а в некоторых случаях и вовсе не работают.

Анализ литературы. Скэшированные данные обычно хранятся в базе данных SQL [2], или каком-либо специализированном хранилище, облегчающем поиск нужной записи. Самый простой вариант, конечно же, просто сохранить в файле на диске, однако тут уже начинаются сложности с хранением каких-либо дополнительных данных о записи, например, времени жизни, частоте посещения.

Кэш не хранится вечно. Если бы вывод страницы никогда не менялся, то это была бы уже не интерактивная страница, и выполнение каких-либо сценариев на ней теряет всякий смысл – это стало бы просто статичной HTML-страницей. Поэтому обычно у записей кэша есть такое понятие, как срок жизни – сколько времени кэш хранится. По истечении этого срока он сбрасывается, чтобы скэшировать вероятно изменившийся вывод сценария. Срок жизни задается обычно от получаса до месяца, в зависимости от частоты изменения содержимого и посещаемости страницы. Чем чаще посещаемость – тем меньшее время жизни можно поставить (если необходимо), чтобы по-прежнему быть в выигрыше. Однако, не всегда можно точно определить сколько времени нужно, чтобы страница изменилась. Примером может послужить интернет-форум, или гостевая книга на каком-либо сайте. Заведомо неизвестно когда кто-нибудь решит оставить там свое сообщение. Поэтому в таких случаях кэш либо не используется, либо разрабатывается специально как неотъемлемая часть страницы. Применять полностью автоматические средства не представляется возможным.

Решение задачи. Разработанный модуль автоматически отслеживает, не изменяется ли вывод кэшируемого фрагмента при одинаковых условиях. Условия – это адрес страницы и параметры, которые ей передаются. Было решено пожертвовать немного автономности разрабатываемого модуля, и предоставить программисту возможность самостоятельно выбрать, какие параметры важны. Программисту даётся возможность самому указать массив с параметрами, которые влияют на результат. Никакая автоматическая система не может знать этого лучше, чем сам разработчик. Времени на это все равно практически не затрачивается.

Таким образом, получился модуль-объект, имеющий всего 2 важные функции: старт и стоп. Программист расставляет их в своем сценарии как метки начала/конца кэшируемых фрагментов. Стартовой функции в качестве параметров передает массив аргументов, влияющих на вывод фрагмента, и получает от нее ответ true либо false, говорящий о том, скэширован фрагмент с такими значениями этих параметров или нет. Если скэширован (true), тогда весь блок можно пропустить до функции конца, которая и выведет сохраненный результат. Если же фрагмент пока не скэширована, то стартовая функция включает буферизацию вывода, и возвращает ответ false. Сценарий фрагмента отрабатывает в обычном режиме, и затем функция конца собирает выведенный текст, и может принять решение о сохранении его в кэше.

Решение проблемы со «временем жизни» КЭШа. Как уже было обозначено выше, кэш обычно хранится только какое-то определенное время, после

которого, предположительно, результат работы блока мог измениться [3]. Однако могло ничего не измениться, и это получится просто дополнительный расход процессорного времени на выполнение блока снова несколько раз., чтобы определить изменяется ли он, целесообразность его хранения в зависимости от соотношения время/объем и прочих условий чтобы в итоге снова сохранить тот же результат, который и был до этого.

В специальных системах кэширования, разрабатываемых специально для конкретных приложений, эта проблема решается путем вызова команды обнуления КЭШа из той же функции, которая, собственно, и изменит вывод страницы. То есть при добавлении/удалении/изменении каких-либо записей в базе данных заведомо известно, на какие страницы это повлияет.

В стандарте SQL 2003-го года появляются триггеры – это SQL процедуры, которые могут выполняться автономно при совершении каких-либо действий над таблицами, с которыми они связаны [4]. Есть два типа триггеров: которые выполняются перед выполнением запроса (BEFORE), и после выполнения запроса (AFTER). И три типа события, на которые они могут реагировать: действие, которое влечет за собой добавление новой строки в таблицу (например, запрос INSERT или REPLACE), изменение данных какой-либо строки (например, запрос UPDATE), и удаление строки из таблицы (например, DELETE). Это дает возможность условного обновления КЭШа страниц. Разработанный модуль позволяет программисту сообщить, какие строки, каких таблиц влияют на страницу. Это дает возможность автоматически создать триггер, который при изменении указанных строк обнуляет и связанную с ними запись КЭШа. Таким образом, больше не нужно угадывать время, за которое страница изменится, а также, это избавляет от необходимости лишних действий – обновления КЭШа, когда на самом деле ничего не изменяется.

Выводы. Разработанный программный модуль кэширования прост в применении (всего 2 функции – метки начала и конца кэшируемого блока). Его внедрение занимает немного времени, однако дает гораздо более значительный прирост производительности, нежели полностью автоматическая система кэширования, а также обладает несколько большей надежностью. В грамотно спроектированном, чётко структурированном веб-приложении разработанный внешний модуль даёт такой же прирост производительности, как и встроенная система кэшированя, разработанная как неотъемлемая часть программы.

Список литературы: **1**. *Система управления* содержимым [электронный ресурс] URL:ru.wikipedia.org/wiki/Система_управления_содержимым **2.** *Язык* структурированных запросов SQL [электронный ресурс] URL: ru.wikipedia.org/wiki/SQL **3**. *Выбор оптимального времени жизни* кэша [электронный ресурс] URL:joomlaportal.ru/content/view/963/68/ **4**. *Триггеры* в СУБД [электронный ресурс] URL:ru.wikipedia.org/wiki/Триггер_(базы_данных)