УДК 004.622

ПОВЫШЕНИЕ СКОРОСТИ ФОРМИРОВАНИЯ СТРАНИЦ WEB-ПРИЛОЖЕНИЯ ПУТЕМ ПРЕОБРАЗОВАНИЯ ЗАПРОСА К БАЗЕ ДАННЫХ В НАБОР ЦИКЛОВ

С. Л. Зиноватная[1], А. А. Зиноватная[1], Н. В. Швец[2]
[1]Одесский национальный политехнический университет
[2]Одесская Национальная Академия Пищевых Технологий

Аннотация. *Представлены формальные правила преобразования запроса для ускорения вывода результирующей таблицы в web-приложении путем устранения операции соединения. Описан алгоритм преобразования запроса на основе изложенных правил. Показано, что время получения результата снижается в 1,7 раза.*

Ключевые слова: *база данных, запрос, цикл, операция соединения.*

Введение

В настоящее время информационные системы (ИС), хранящие нужные факты в базе данных (БД), применяются практически во всех областях жизни. Несмотря на возникновение новых технологий хранения данных, по-прежнему использование реляционных БД является доминирующим [1-4]. Большое количество ИС реализовано в виде web-приложений. Независимо от того, каким образом реализована ИС, в современном мире для пользователя важно максимально быстро получать данные на экран. В ИС зачастую экранные формы формируются на основе информации, извлеченной из БД. Поэтому узким местом может стать скорость выполнения запроса, особенно, если запрос обращается к большому количеству таблиц, и эти таблицы имеют большой размер.

Существуют различные способы повышения производительности запросов. Один из способов – использование индексов в таблицах [5]. Однако настройка индексов требует внимательности и специального изучения работы системы, поскольку при частом обновлении данных индексы могут замедлять работу системы [6]. Можно изменить структуру БД, нарушив правила нормализации, однако это тоже может привести к ухудшению производительности [7]. Настройка параметров СУБД [8] требует, во-первых, от разработчика знаний тонкостей работы системы управления базами данных (СУБД), а, во-вторых, иногда программист может не иметь доступа к настройкам СУБД. В некоторых случаях может помочь использование материализованных представлений, что также требует изучить список запросов, поступающих в БД за значительный период [9].

В [10] названы причины, почему могут использоваться циклы в коде программы с запросами SQL: наличие логики; необходимость изменить операции, которые последовательно влияют на несколько таблиц; ситуация, когда разработчик хочет, чтобы изменения регулярно фиксировались внутри цикла. Отмечено, что во многих случаях можно упростить сложную процедурную логику и перенести ее в операторы SQL, чтобы устранить циклы в программе. В остальных случаях предлагается убирать циклы. Но в качестве примеров приведены запросы, обращающиеся к одной таблице.

Количество различных способов повышения производительности ИС и количество формальных и неформальных советов, как это сделать, показывает, что задача повышения скорости выполнения запросов является актуальной.

В большом количестве приложений БД состоит из значительного количества взаимосвязанных таблиц. Для таких БД можно получить нужную информацию, которая хранится в различных таблицах, написав единственный запрос. Но такой запрос будет содержать большое количество соединений.

В СУБД существуют специальные алгоритмы оптимизации выполнения запросов с соединениями [11]. Однако может получиться, что конкретная СУБД не имеет таких алгоритмов, либо используются операции левого (правого) соединения, для которых алгоритмы не работают.

В [12, 13] описаны варианты распределенной обработки запросов и транзакций, которые предполагают использование новейших сетевых технологий и технологий параллельных вычислений.

Предлагаются новые разновидности СУБД для решения задачи ускорения выполнения запросов [14].

В реальной жизни существует большое количество ИС в организациях, которые не обладают большими финансовыми ресурсами. Зачастую система работает внутри организации с использованием локальной сети. ИС может работать продолжительное время, пока количество хранящихся в ней записей не станет критическим с точки зрения скорости обслуживания клиентов компании. В таких случаях, когда невозможно изменить аппаратную составляющую ИС, изменить СУБД и структуру БД, полезным является наличие средств для автоматизации преобразования уже имеющегося кода, что даст возможность быстро и без значительных затрат получить существенное уменьшение времени отклика системы на запрос пользователя.

Целью работы является уменьшение времени формирования страниц для вывода результатов запроса за счет преобразования текста запроса с использованием формального алгоритма для получения последовательности циклов, обрабатывающих сформированные части запроса.

1. Представление запроса

Представим запрос в следующем виде:

$$q=<T, S, W, O, R>,$$

где T - множество таблиц, участвующих в запросе, каждый элемент множества T представляет собой кортеж $< nt, F, pk >$, где nt – имя таблицы; F – множество полей таблицы; pk – первичный ключ таблицы;

S - множество полей, которые выводятся в результирующую таблицу, в S можно выделить подмножества STi, содержащие поля, которые принадлежат таблице Ti или могут быть вычислены на основании таковых, $i \le |T|$;

W - множество полей, использованных в предложении where;

O – множество полей, по которым выполняется сортировка:

R – множество связей между таблицами множества T, каждый элемент множества R представляет собой кортеж $<pt, ft>$, где pt – родительская таблица; ft – дочерняя таблица, $pt \in T$, $.ft \in T$.

Каждый элемент множеств S, W, O представляется в виде $t.f$, где t – таблица, $t \in T$, или набор таблиц, если поле вычисляется на основе полей из нескольких таблиц f - имя поля.

2. Преобразование запроса для формирования частичных запросов

Рассмотрим ситуацию, когда результат запроса выводится в виде форматированной таблицы, программа в цикле проходит все строки результирующей таблицы tq и формирует строки для вывода (рис. 1).

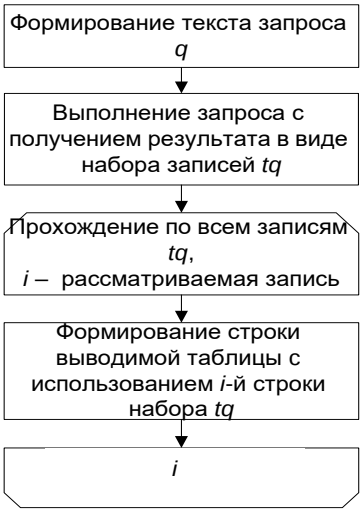


Рис. 1 Схема алгоритма формирования таблицы для вывода на экран или печать с помощью единого запроса

Можно разбить запрос q на несколько запросов, чтобы устранить ресурсоемкие операции соединения. Возможен вариант, при котором все циклы для просмотра наборов строк, полученных в результате выполнения частичных запросов, вложены полностью друг в друга (рис. 2) либо результаты отдельных частичных запросов могут быть выполнены с одноразовым просмотром результата (рис. 3).

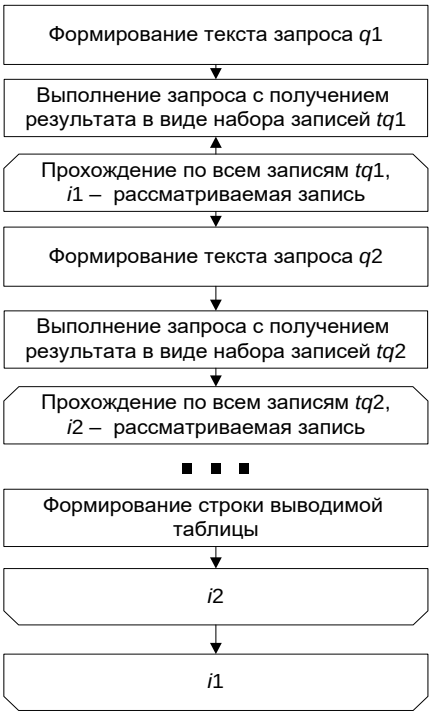


Рис.2 Схема алгоритма формирования таблицы для вывода на экран или печать в случае разбиения запроса с организацией вложенных циклов

Удобно иметь алгоритм, который автоматизирует формирование множества частичных запросов *qi*, поскольку не каждый запрос можно преобразовать подобным образом.

Для определения варианта формирования полного вложения циклов или варианта с циклами одноразового просмотра результата необходимо сформировать с использованием множества *R* иерархию таблиц запроса *q* *T*1→*T*2→*T*3…, при этом родительская таблица всегда имеет номер меньше, чем дочерняя таблица.

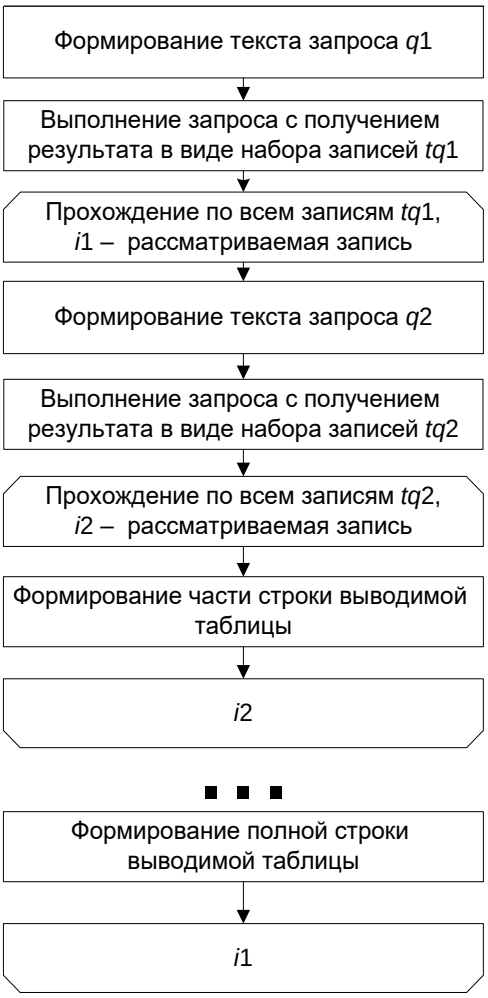


Рис.3 Схема алгоритма формирования таблицы для вывода на экран или печать в случае разбиения запроса с организацией частично вложенных циклов

Рассмотрим варианты запроса *q* с учетом участия в предложении where первичного ключа таблицы: а – первичные ключи отсутствуют; б – присутствует ключ родительской таблицы.

Примером подобных вариантов служат следующие запросы (в результирующую таблицу включены поля из родительской и из подчиненной таблицы): а – список всех заказов всех клиентов; б – список всех заказов конкретного клиента.

Вариант А. Если в предложении where отсутствует условие, в котором первичный ключ родительской таблицы *pt*1 сравнивается с константой, то можно сформировать запрос *qt*2, который получен из *q* исключением операции соединения с таблицей *T*1; затем выполнить *qt*2, в цикле просмотра его результирующего набора выполнить запрос *qt*1 для определения полей множества *ST*1, который обращается только к таблице *T*1, и сформировать строку результирующей таблицы с использованием *ST*1 (рис. 4).

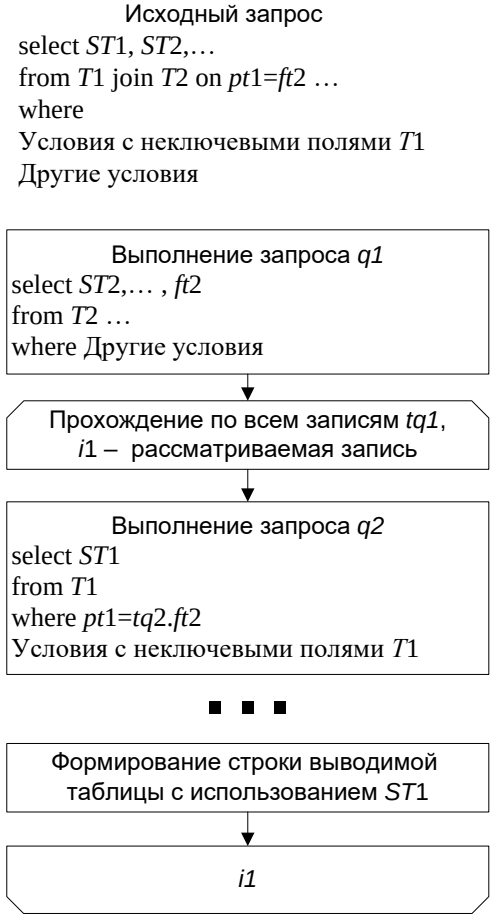


Рис.4 Схема алгоритма преобразования запроса, если первичные ключи отсутствуют в предложении where

Вариант Б. Если в предложении where присутствует условие, в котором первичный ключ родительской таблицы *pt*1 сравнивается с константой, то можно сформировать запрос *qt*1 для определения полей множества *ST*1, который обращается только к таблице *T*1, и содержащий условие с первичным ключом *T*1, выполнить его до выполнения запроса *qt*2, который получен из *q* исключением операции соединения с таблицей *T*1; затем выполнить *qt*2 и сформировать строку результирующей таблицы с использованием *ST*1 (рис. 5).

Такое преобразование возможно, если выполнены следующие ограничения:

- в *ST*1 отсутствуют поля, у которых t включает больше одной таблицы, то есть в *S* нет вычисляемых полей, которые вычисляются с использованием *T*1 и других таблиц;
- в множестве *O* отсутствуют поля из *ST*1, $O \cap ST1 = \varnothing$.

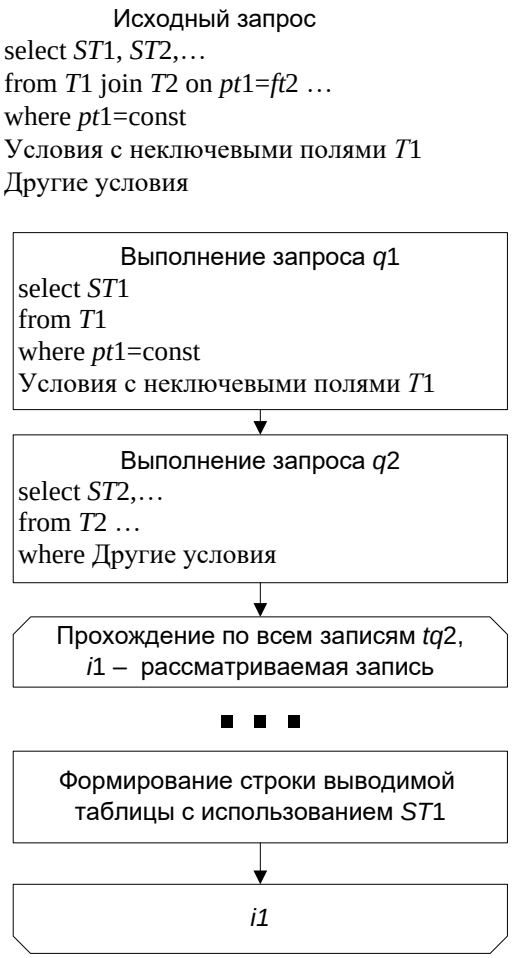


Рис.5 Схема алгоритма преобразования запроса, если первичные ключи присутствуют в предложении where

Если в результате преобразования таблица удалена из множества *T* исходного запроса после устранения операции соединения, то необходимо заменить ключевые поля этой таблицы, присутствующие в множестве *S*, на поля, соответствующие внешним ключам дочерней таблицы, соединение с которой было устранено. Если ключевые поля отсутствуют во множестве *S*, то поля, соответствующие внешним ключам дочерней таблицы, соединение с которой было устранено, следует добавить в *S*.

Такие действия могут быть применены последовательно для каждой пары $Ti \rightarrow Ti+1$ в иерархии таблиц запроса *q*.

Таким образом, общий алгоритм преобразования запроса состоит из следующих шагов.

1. С помощью метаданных и парсинга запроса получить *T, S, W, O, R*.
2. Сформировать иерархию таблиц.
3. $i=1$.
4. Рассмотреть пару $Ti \rightarrow Ti+1$.
5. Если $O \cap STi \neq \varnothing$, перейти в п.8.
6. Если в *STi* присутствуют поля *f*, у которых *t* содержит более одной таблицы, перейти в п.8.
7. Если предложение where содержит условие *pti*=const, то выполнить преобразование запроса *q* по варианту А. Иначе выполнить преобразование по варианту Б.
8. $i = i+1$.
9. Если $i < |T|$, то перейти в п.4.
10. Выполнить изменение кода программы в зависимости от варианта преобразования запроса. Конец алгоритма

3. Эксперимент

Для проверки результата исследования проведен эксперимент с использованием программы учета деятельности медицинского центра. ИС интенсивно используется в течение рабочего дня для обслуживания клиентов, предполагает запись на прием и для проведения различных видов исследования, просмотр результатов оказания услуг, а также ведение статистики. Система реализована в локальной сети, интерфейс создан с использованием Java Script и PHP, СУБД MySQL. Таким образом, использованы достаточно часто применяемые инструменты для реализации ИС относительно небольших организаций [15].

Как уже сказано, одной из функций программы является получение статистики выполненных услуг за указанные период времени (рис. 6), скрыты личные данные пациентов.

Рис. 6 Форма вывода статистических данных

Данный запрос в системе является не единственным кандидатом для применения описанного в статье алгоритма. Этот запрос выбран для примера, поскольку демонстрирует типичный вариант соединения набора таблиц для получе-

ния результирующей таблицы, которая содержит неключевые поля нескольких таблиц.

В частности, в примере используется запрос *q* следующего вида:

```
SELECT p.nameplace, pl.dateplaner, s.idservice,
pc.idpersonour,
concat(pc.name1person, ' ', pc.name2person , '
',pc.name3person) as namep,
concat(e.name1employee,'
',substring(e.name2employee,1,1),'.',substring(e.n
ame3employee,1,1),'.') as namee,
s.nameservice as n, psm.price, pl.comment,
pl.prescription, pc.addcomment, psm.q,
psm.isinlog, psm.idserviceplaner, pl.oplata,
pl.dateoplata, psm.numberinlog,
cd.closeddate,psm.idemployeeforreport,
pl.idemployee,s.idlog, psm.idfilial,
filial.namefilial, namelog,
concat(e2.name1employee,'
',substring(e2.name2employee,1,1),'.',substring(e
2.name3employee,1,1),'.') as namee2
FROM serviceplaner psm join planer pl
join person pc join places p
join service s join employee e2
join employee e join closeddays cd
join filial join logs
on s.idservice=psm.idservice and
pl.idplaner=psm.idplaner
and pc.idpersonour=pl.idperson and
p.idplace=pl.idplace
and psm.idemployeeforreport=e.idemployee
and pl.idemployee=e2.idemployee and
pl.dateplaner=cd.closeddate
and cd.idemployee=psm.idemployeeforreport and
s.idlog=cd.idlog
and psm.idfilial=cd.idfilial and fil-
ial.idfilial=psm.idfilial and logs.idlog=cd.idlog
where dateforreport between '2018-12-01' and
'2018-12-31' and pl.deletedrow=0
order by psm.dateforreport, psm.numberinlog,
p.nameplace
```

В результате анализа запроса получена иерархия таблиц, показанная на рис.7.

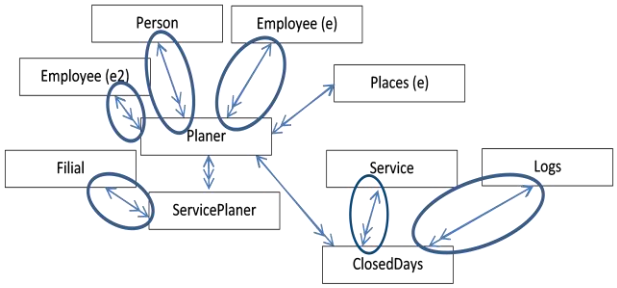


Рис.7 Иерархические связи между таблицами запроса

Для обведенных связей выполнено преобразование запроса согласно вышеизложенному алгоритму. Получены следующие запросы:

```
q1: SELECTp. nameplace, dateplaner,
psm.idservice,pl.idperson , psm.price,
pl.comment,pl.prescription,q,isinlog,
psm.idservice,
psm.idserviceplaner,pl.oplata,pl.dateoplata,
psm.numberinlog,
cd.closeddate,psm.idemployeeforreport,
pl.idemployee,s.idlog,psm.idfilial
FROM serviceplaner psm join planer pl
join places p join closeddays cd
on pl.idplaner=psm.idplaner and
p.idplace=pl.idplace and
pl.dateplaner=cd.closeddate
and cd.idemployee=psm.idemployeeforreport
and s.idlog=cd.idlog and psm.idfilial=cd.idfilial
where dateforreport between '2018-12-01' and
'2018-12-31' and pl.deletedrow=0
order by dateforreport,numberinlog,nameplace
q2: SELECT concat(pc.name1person,'
',pc.name2person , ' ',pc.name3person) as namep,
pc.addcomment
FROM  person pc where
pc.idpersonour=.$m['idperson']
q3: SELECT concat(e.name1employee,'
',substring(e.name2employee,1,1),'.',substring(e.n
ame3employee,1,1),'.') as namee
FROM  employee e where
e.idemployee=.$m['idemployee']
q4: SELECT concat(e.name1employee,'
',substring(e.name2employee,1,1),'.',substring(e.n
ame3employee,1,1),'.') as namee2
FROM  employee e where
e.idemployee=".$m['idemployeeforreport']
q5: select namelog from logs where
idlog=.$m['idlog'];
q6: SELECT namefilial  from filial p where idfil-
ial=.$m['idfilial']
q7: SELECT nameservice  from service s where id-
service=.$m[idservice]
```

Для связи между таблицами Places и Planer, преобразование невозможно, так как неключевое поле таблицы Places использовано в списке сортировки. Аналогично для пары Planer и Service-Planer.

Для эксперимента использованы две БД: БД за 2016 год (архивная БД, содержит записи только за 2016 г.) и 2018 год (содержит записи за 2018 год и еще неперемещенные в архив записи за 2017 г.). Исходные таблицы запроса содержат различное количество строк в различных БД (табл.1).

Очевидно, что с ростом количества данных растет время выполнения запроса. Рассмотренный в качестве примера запрос демонстрирует получение статистики. Однако в ИС существуют и другие запросы, требующие соединения нескольких таблиц, например, для получения списка анализов конкретного пациента. Низкое время

выполнения таких запросов приводит к ухудшению времени обслуживания пациентов, что критично для любой коммерческой организации.

Таблица 1

Размер таблиц БД, использованных в эксперименте

Таблица	Количество строк	
	БД 2016	БД 2018
person	19375	26343
service	403	500
planer	18745	65984
serviceplaner	19394	68254
places	17	16
employee	46	52
closeddays	1945	3907
filial	13	14
logs	2	2

Следует отметить, в данной БД уже выполнена денормализация структуры, что уже снизило время выполнения запросов. Однако данная операция приводит к сложности в последующем обслуживании БД, поскольку система является постоянно развивающейся и требует внедрения новых функций по желанию владельца.

Измерено время получения результата, содержащего различное количество строк. Для этого использованы разные периоды для задания условия отбора строк (табл. 2, 3).

Таблица 2

Время выполнения запросов для БД 2016 г.

Период	Количество строк в таблице-результате	Для исходного запроса, сек.	Для набора циклов, сек.
2 недели	39	1,62	0,73
1 месяц	79	2,94	1,38
2 месяца	133	5,25	2,95
3 месяца	182	7,5	3,75
4 месяца	248	10,13	4,79
6 месяца	311	14,80	6,92

Таблица 3

Время выполнения запросов для БД 2018 г.

Период	Количество строк в таблице-результате	Для исходного запроса, сек.	Для набора циклов, сек.
2 недели	668	1,39	1,18
1 месяц	1128	16,01	6,92
2 месяца	2493	34,56	16,90
3 месяца	4127	60,73	32,91
4 месяца	5268	67,93	47,93
6 месяца	6526	95,54	61,33

На рис.8 и рис. 9 полученные результаты показаны в виде графика.

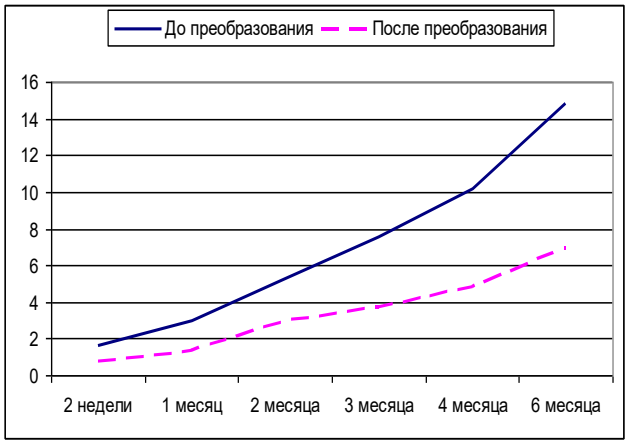


Рис.8 Время получения результата для БД 2016 г.

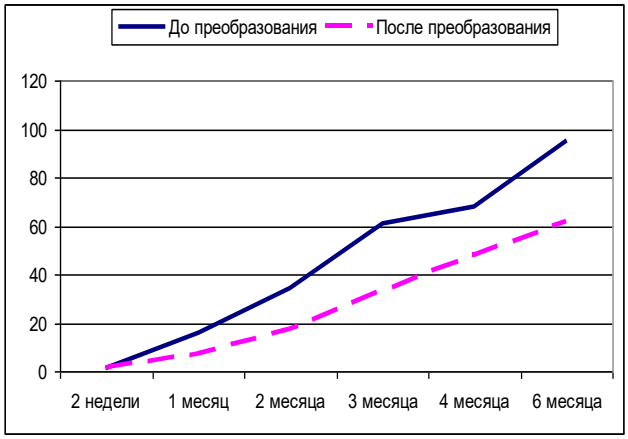


Рис.9 Время получения результата для БД 2018 г.

Данный эксперимент подтвердил, что выполненное по предложенным правилам преобразование запроса уменьшает время отклика на запрос пользователя.

Выводы

Формализованы правила преобразования запроса для ускорения вывода результирующей таблицы в web-приложении путем устранения операции соединения. Для этого предложено представление запроса в виде множественной модели. Описан алгоритм преобразования на основе изложенных правил.

Показано, что время получения результата может быть снижено существенно для БД, в которой количество записей в таблицах порядка десятка тысяч, и при этом запрос требует соединения нескольких таблиц для получения результата.

В дальнейшем могу быть рассмотрены алгоритмы автоматизированного изменения исходного текста программы для внедрения полученных частичных циклов.

Список использованной литературы

1. Рейтинги сервисов и технологий [Электронный ресурс]. – Режим доступа: https://tagline.ru/database-management-systems-rating/

2. Top 30 Most Popular Database Management Software: Complete List [Electronic Resource]. – Access Mode: https://www.softwaretesting help.com/database-management-software/

3. Top 10 databases you should learn in 2019 [Electronic Resource]. – Access Mode: https://www.improgrammer.net/top-10-databases-should-learn-2015/

4. Software Testing Popular Database Management Systems [Electronic Resource]. – Access Mode: https://www.gcreddy.com/ 2016/09/ popular-database-management-systems.html

5. Кунгурцев, А., Зиноватная, С., Мусанна, С., Манчук, А. Анализ запросов к базе данных для определения полей-кандидатов на создание кластерного индекса [Текст]. *Сборник доклади от юбилейна научна конференция с международно участие «Автоматика, управление и информационно технологии».* Том I. Варна. 2010. С. 127–130.

6. Зиноватная, С. Л. Левченко, А. Ю., Галанюк, К. А. Имитационная модель для проектирования оптимальной индексной структуры базы данных [Текст]. *Радіоелектронні і комп'ютерні системи*. Харків, «ХАІ», 2013. №5 (64). С.255–260.

7. Зіноватна, С. Л. Метод формування первинної множини варіантів денормалізації на основі аналізу запитів до бази даних [Текст]. *Вісник Чернігівського державного технологічного університету.* Збірник. Чернігів : ЧДТУ, 2009. №37. С. 231–238.

8. Barthels, C., Alonso, G., Hoefler, T. Designing Databases for Future High-Performance Networks [Text]. IEEE Data Engineering Bulletin. Vol. 40, No. 1. 2017.

9. Кунгурцев, А. Б., Возовиков, Ю. Н. Поддержка эффективности механизма управления материализованными представлениями [Текст]. *Електротехнічні та комп'ютерні системи.* 2011. №4. С. 136–140.

10. Wandschneider, M. Core Web Application Development with PHP and MySQL. Wandschneider [Text]. Prentice Hall PTR, 2005. 912 p.

11. Коннолли, Т., Бегг, К., Страчан, А. Базы данных: проектирование, реализация и сопровождение. Теория и практика [Текст]. 3-е изд. Москва: Изд. дом "Вильямс". 2003. 1440 с.

12. Smith, G. PostgreSQL 9.0 High Performance [Text]. Packt Publishing Ltd. 2010. 442 p.

13. Taniar, D. High Performance Database Processing [Text]. IEEE 26th International Conference on Advanced Information Networking and Applications. 26-29 March. 2012. P. 5–6.

14. Insoon Jo, Duck-Ho Bae, Andre S. Yoon, Jeong-Uk Kang, Sangyeun Cho, Daniel D. G. Lee, Jaeheon Jeong: YourSQL. A High-Performance Database System Leveraging In-Storage Computing [Text]. Proceedings of the VLDB Endowment. 9(12). 2016. P. 924–935.

15. Фаро, С., Паскаль, Л. Рефакторинг SQL-приложений [Текст]. Пер. с англ. СПб: Символ-Плюс. 2009. 336 с.

References

1. "Service and technology ratings" [Rejtingi servisov i texnologij], avaliable at: https://tagline.ru/database-management-systems-rating/

2. "Top 30 Most Popular Database Management Software: Complete List", avaliable at: https://www.softwaretestinghelp.com/database-management-software/

3. "Top 10 databases you should learn in 2019", avaliable at: https://www.improgrammer.net/top-10-databases-should-learn-2015/

4. "Software Testing Popular Database Management Systems", avaliable at: https://www.gcreddy.com/2016/09/popular-database-management-systems.html

5. Kungurtsev, A. B., Zinovatnaya, S., Musanna, S., Manchuk, A. (2010) "Analysis of queries to the database for determination of fieldscandidates on creation of cluster index" [Analiz zaprosov k baze dannyh dlya opredeleniya polej-kandidatov na sozdanie klasternogo indeksa]. Collection of lectures of anniversary scientific conference with international participation «Automation, management and information technologies». Varna. Vol. I. P. 127–130.

6. Zinovatna, S. L., Levchenko, O. Yu., Halaniuk, K. O. (2013) "The simulation model for designing the optimal index structure of the database" [Imitacionnaya model dlya proektirovaniya optimalnoj indeksnoj struktury bazy dannyh]. Electronic and computer systems. «HAI». №5 (64). P.255–260.

7. Zinovatna, S. L. (2009) "Method of forming the primary set of denormalization variants based on the analysis of database queries" [Metod formuvannya pervinnoї mnozhini variantiv denormalizacії na osnovi analizu zapitiv do bazi danih]. Bulletin of Chernihiv state technological University. Collector. «CSTU»№ 37. P. 231–238.

8. Barthels, C., Alonso, G., Hoefler, T. Designing Databases for Future High-Performance Net-

works. IEEE Data Engineering Bulletin. Vol. 40, No. 1. 2017.

9. Kungurtsev, A. B., Vozovikov, Y. N. (2011) "Support of efficiency of management mechanism by materializes views" [Podderzhka effektivnosti mexanizma upravleniya materializovannymi predstavleniyami]. Electrical engineering and computer systems. № 04 (80). P. 136–140.

10. Wandschneider, M. Core Web Application Development with PHP and MySQL. Wandschneider. Prentice Hall PTR, 2005. 912 p.

11. Konnolli, T., Begg, K., Strachan, A. (2003) "Connolly T., Begg C., Strachan A. Databases: design, implementation and maintenance. Theory and practice" [Bazy dannyh: proektirovanie, realizaciya i soprovozhdenie. Teoriya i praktika]. Moscow. 1440 p.

12. Smith, G. PostgreSQL 9.0 High Performance. Packt Publishing Ltd. 2010. 442 p.

13. Taniar, D. High Performance Database Processing. IEEE 26th International Conference on Advanced Information Networking and Applications. 26-29 March. 2012. P. 5–6.

14. Insoon Jo, Duck-Ho Bae, Andre S. Yoon, Jeong-Uk Kang, Sangyeun Cho, Daniel D. G. Lee, Jaeheon Jeong: YourSQL. A High-Performance Database System Leveraging In-Storage Computing. Proceedings of the VLDB Endowment. 9(12). 2016. P. 924–935.

15. Pharo, S., Pascal, L. (2009) "Refactoring SQL Applications" [Refaktoring SQL prilozhenij]. SPb. 336 p.

IMPROVE THE SPEED OF WEB APPLICATION PAGE GENERATION BY TRANSFORMING A DATABASE QUERY INTO A SET OF LOOPS

S. L. Zinovatna[1], H. O. Zinovatna[1], N. V. Shvets[2]
[1]Odessa National Polytechnic University
[2]Odessa National Academic Food Technologies

Abstract. *Currently, despite the emergence of new storage technologies, the use of relational databases is dominant. In almost all areas of human life there are information systems with such databases, and often they are implemented in the form of web-applications. It is important to quickly receive a response from the user to his request. However, in the case of a complex system, the query can access a large number of tables, which will take a long time. There are various ways to speed up the information system: indexing, denormalization, the use of materialized views, configuring the DBMS parameters. However, the use of such methods to improve system performance requires careful study and analysis of the system for a long period of time, otherwise you can get the opposite result. The paper proposes a formal way to transform the query text with the selection of partial queries in it to execute them inside or outside the cycle of viewing query records in the formation of the resulting table. The view of the query, which formalizes the rules of transformation of the text of the original query in order to eliminate resource-intensive connection operations, is considered. The algorithm of transformation of query for use of partial queries is presented. Various options for using key and non-key attributes from the tables that are accessed by the query are described, which determines the type of transformation of the query. It is shown how the time of obtaining the result after the performed transformations is reduced, by the example of a real system of a medical institution, which is based on a database with a large number of tables related to each other, and requires a large number of queries per unit of time, while the queries refer to many tables.*

Keywords: *database, query, loop, join operation.*

ПІДВИЩЕННЯ ШВИДКОСТІ ФОРМУВАННЯ СТОРІНОК WEB-ЗАСТОСУВАННЯ ШЛЯХОМ ПЕРЕТВОРЕННЯ ЗАПИТУ ДО БАЗИ ДАНИХ У НАБІР ЦИКЛІВ

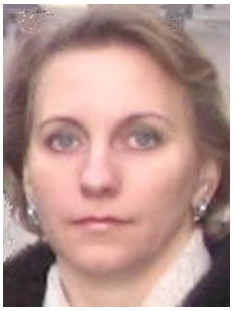
С. Л. Зіноватна[1], Г. О.Зіноватна [1], Н. В. Швець[2]
[1]Одеський національний політехнічний університет
[2]Одеська Національна Академія Харчових Технологій

Анотація. *Розглянуті причини повільного виконання запитів до бази даних, які вимагають звертання до великої кількості таблиць. Описані засоби зменшення часу отримання відгуку користувачем на свій запит від інформаційної системи, в основі якої лежить реляційна база даних. Проаналізовані причини неможливості застосування таких засобів в умовах деяких реальних систем. Надане формальне представлення запиту до бази даних, яке дозволяє виконати розподілення одного запиту*

на декілька окремих запитів для подальшої обробки. Описано загальний алгоритм отримання результуючої таблиці як результату виконання запиту в застосуванні, яке звертається до бази даних. Представлено формальні правила перетворення запиту для прискорення виведення такої результуючої таблиці для web-застосування шляхом усунення операції з'єднання на основі запропонованого формального представлення. Описані різні варіанти використання ключових та неключових атрибутів з таблиць, до яких звертається запит, що визначає вид перетворення запиту. Надано алгоритм перетворення запиту на основі викладених правил, в процесі якого отримуються часткові запити, для кожного з яких потрібно створити цикл для перегляду результатів запиту. Описано, яким чином такі цикли можуть бути вбудовані в код програми для отримання найкращого ефекту. Визначено, в яких випадках можливе винесення виконання часткових запитів за рамки виконання циклу обробки записів основного запиту. Показано, як знижується час одержання результату після виконаних перетворень, на прикладі реальної системи медичної установи, яка має в основі базу даних з великою кількістю таблиць, пов'язаних між собою, та вимагає виконання великої кількості запитів одиницю часу, при тому, що запити звертаються до багатьох таблиць.

Ключові слова: *база даних, запит, цикл, операція з'єднання, час виконання запиту.*

Получено 15.02.2018



Зиноватная Светлана Леонидовна, кандидат технических наук, доцент, доцент кафедры системного программного обеспечения Одесского национального политехнического университета. Просп. Шевченко, 1, Одесса, Украина, E-mail: zinovatnaya.svetlana@opu.ua, тел. +38-067-939-00-94

Svitlana Zinovatna, Ph.D., associate Professor, associate Professor of system software, Odessa National Polytechnic University, Shevchenko ave., 1, Odessa, Ukraine, E-mail: zinovatnaya.svetlana@opu.ua

ORCID ID: 0000-0002-9190-6486

Зиноватная Анна Александровна, магистрант кафедры системного программного обеспечения Одесского национального политехнического университета. Просп. Шевченко, 1, Одесса, Украина, E-mail: zinovatnaya@gmail.com, тел. +38-096-710-08-29

Hanna Zinovatna undergraduate of system software, Odessa National Polytechnic University, Shevchenko ave., 1, Odessa, Ukraine, E-mail: zinovatnaya@gmail.com

ORCID ID: 0000-0001-9647-5171

Швец Наталья Васильевна, старший преподаватель кафедры информационных технологий и кибербезопасности Одесской национальной академии пищевых технологий. Ул. Канатная, 112, Одесса, Украина, E-mail: shvetsnv0601@gmail.com, тел. +38-067-947-65-53

Natalya Shvets, Senior Lecturer, Department of Information Technology and Cyber Security, Odessa National Academy of Food Technologies, Kanatnaya str., 112, Odessa, 65039, Ukraine, E-mail: shvetsnv0601@gmail.com

ORCID ID: 0000-0002-6719-3842