Использование методов оптимизации запросов к распределенной базе данных для повышения эффективности информационных систем

Т. А. Федосеенко

Московский технологический институт
119334, Москва, Ленинский проспект, 38а
e-mail: velliar@gmail.com

Аннотация. Статья посвящена решению задачи повышения эффективности распределенных баз данных. Для увеличения производительности выполнения запросов необходимо искать решения, анализируя процесс выполнения запросов сервером — заменять запросы с выборкой без повторений, использовать индексы, оптимизировать структуру базы данных на этапе физического проектирования.
Ключевые слова: базы данных, оптимизация запросов, распределенные базы данных.

1. Введение

Оптимизация является одним из важных направлений разработок в области исследований баз данных (БД). Согласно Джону Бэнтли, под оптимизацией понимается модификация системы для улучшения ее эффективности [1]. Хорошо оптимизированная БД значительно снижает нагрузку на сервер, и увеличивает общую производительность [3, 4].

В статье рассматривается информационная система, реализованная на основе СУБД Access, обладающей следующими достоинствами: БД способна обеспечить достаточную производительность при работе с системой порядка сотни пользователей с единой базой данных на сервере; обладает невысокой стоимостью; имеет интуитивно понятный и простой способ установки на клиентские компьютеры; имеет в составе средства, позволяющие обеспечить формирование третьей нормальной формы объектов реляционной базы данных; обеспечивает поддержку русском языке и интегрирована с продуктами Microsoft Office.

2. Постановка задачи

Рассматриваемая база данных, логическая структура которой представлена на рис. 1, представляет собой базу данных, приведенную к третьей нормальной форме.

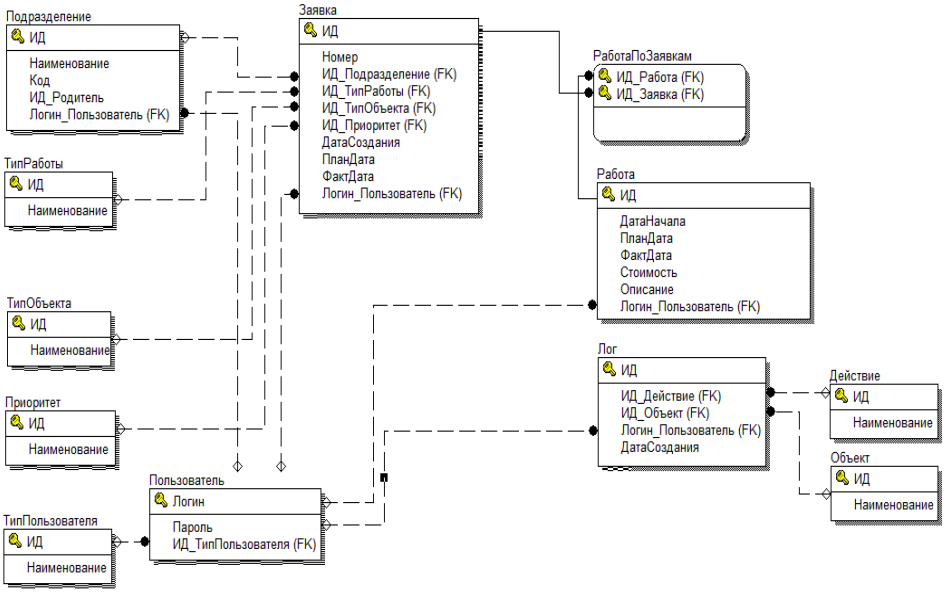

Рисунок 1. Структура БД

То, что БД находятся в третьей нормальной форме, подразумевает, что они находятся и в первой и второй нормальной форме [4, 5]. Согласно первой нормальной форме в БД значения столбцов являются атомарными и все записи — уникальны. Отношения находятся во второй нормальной форме и каждый их неключевой атрибут неприводимо зависим от первичного ключа. В третьей нормальной форме отсутствуют транзитивные функциональные зависимости неключевых атрибутов от ключевых, т. е. все неключевые атрибуты зависят только от первичного ключа.

В нормализованной базе данных уменьшается вероятность возникновения ошибок, уменьшается противоречивость хранимой в базе данных информации.

Эффективность системы может оцениваться различными критериями. Проблема оптимизации запросов базы данных составляется из трех основных компонентов [6, 7]:

- оптимизация запросов;
- оптимизация структуры;
- оптимизация сервера.

3. Повышение эффективности на основе анализа запросов

Критерии, определяющие степень эффективности запроса, формируются на основании основных характеристик системы. Применительно к исследуемой информационной системы, с учетом области применения выделим следующие критерии:

- длительность выполнения запросов выборки;

– нагруженность сервера в зависимости от количества запросов в единицу времени;
– длительность выполнения обновления и изменения данных

С учетом специфики системы этих критериев достаточно для оценки эффективности выполнения запросов в системе.

Проанализируем таблицу «Заявки». Будем выполнять простой запрос на выборку всех данных из таблицы «Заявки», меняя общее количество записей в ней. В результате будем фиксировать время выполнения запросов.

Полученные результаты сведены в гистограмму, изображенную на рис. 2. Из диаграммы видно, что при увеличении количества записей в таблице увеличивается и скорость увеличения длительности загрузки.

Рисунок 2. Гистограмма результатов анализа длительности выполнения простых запросов выборки из таблицы с разным числом записей

Проанализируем длительность выполнения запроса на выборку одной записи (при выборке с условием на уникальный идентификатор). Сначала используем для анализа две записи, одна из которых находится условно в начале таблицы, то есть имеет небольшое значение уникального идентификатора и была добавлена одной из самых первых и записи, которая находится в конце и была добавлена одной из последних. Как видно по результатам, время выборки примерно одинаково и невелико. При этом отбор производится по ключевым индексированным полям.

Проверим результат при выборке по этим же записям, но сделав поля, по которым производится отбор, неключевыми и не индексированными. Как видно, время выполнения запросов увеличилось, т. е. использование индексов существенно влияет на производительность выполнения запросов. Проведем аналогичный анализ на строковых данных, осуществив выборку сначала по индексированным полям, затем по полям, не имеющим индексов.

По данным таблицы можно сделать вывод о том, что в случае, если по какому-либо полю часто выполняется выборка, то для обеспечения необходимой производительности нужно создавать для этого поля индексы. Это ускорит поиск в базе данных и длительность выборки.

Степень влияния индексов на производительность в рассмотренном примере отображена на диаграмме на рис. 3 и в табл.

Таблица. Результаты проведенных экспериментов по длительности выборки данных из таблицы «Заявки»

Тип отбираемого атрибута	Неиндексированные поля		Индексированное поле	
	ИД близко к началу	ИД близко к концу	ИД близко к началу	ИД близко к концу
числовой	0,078 с	0,063 с	0,015 с	0,015 с
строковый	0,031 с	0,032 с	0,016 с	0.015 с

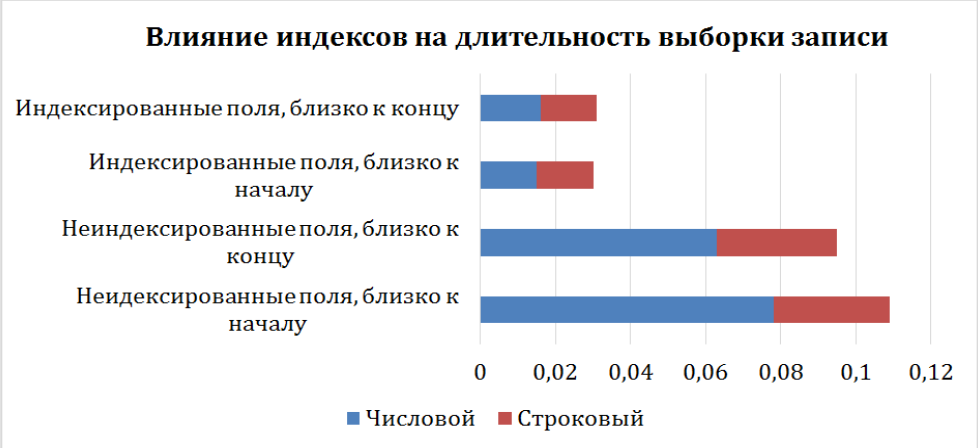

Рисунок 3. Степень влияния наличия индексов на длительность выборки данных из таблицы «Заявки».

Предположим, что необходимо осуществить выборку из заявок всех подразделений, для которых какие-либо заявки были созданы. В этом случае можно воспользоваться запросом

```
select
```

```
  п.Наименование
from
  Заявка з,
  Подразделение п
where
  п.ИД = з.ИД_Подразделение
```

Однако в этом случае мы получим дубликаты наименований подразделений, так как заявки для них могли создаваться неоднократно. Поэтому для исключения повторений наименований используем distinct.

```
select distinct
  п.Наименование
from
  Заявка з,
  Подразделение п
where
  п.ИД = з.ИД_Подразделение
```

Длительность выполнения запроса составляет 0,187 с, что достаточно велико при выборке всего двух записей из базы данных. Попробуем оптимизировать запрос. Прежде всего, проанализируем процесс выполнения запроса на сервере баз данных.

Изначально происходит выборка из двух таблиц: Заявка и Подразделение. Запрос с условием «where» сервер баз данных преобразует в запрос, содержащий «join» — операцию соединения таблиц, которая является подмножеством декартова произведения таблиц. Декартово произведение *n* таблиц — это таблица, содержащая все возможные строки *r*, такие, что *r* является сцеплением какой-либо строки из первой таблицы, строки из второй таблицы, ... и строки из *n*-й таблицы. Таким образом, на первом этапе сервером выполняется декартово произведение таблиц Заявка и Подразделение. Затем происходит выборка данных по условиям. И только на следующем этапе уже после выполнения выборки происходит ее просмотр и исключение дубликатов. Очевидно, что узким местом тут является выборка лишних записей, которые отсеиваются лишь на последнем этапе. Следовательно, нужно изменить синтаксис запроса так, чтобы сначала происходила выборка подразделений сразу без учета тех, для которых не созданы заявки. Это можно сделать с помощью ключевого слова exists.

```
select
  п.Наименование
from
  Подразделение п
where
  exists(
    select
      1
from
    Заявка з
```

```
where
    з.ИД_Подразделение = п.ИД)
```

Измененный запрос будет выглядеть так (оптимизированный запрос). Время выполнения этого запроса составляет всего 0,015 с. Таким образом, заменив запрос с distinct аналогичным запросом с «exists», мы добились уменьшения длительности получения результатов более чем в 12 раз за счет анализа последовательности выполнения запроса на сервере.

Для достижения оптимальной структуры базы данных необходимо соблюдать нормализацию, по возможности на этапе физического проектирования подбирать для атрибутов типы данных, занимающие минимум объема памяти, предпочтительнее выбирать числовые типы вместо строковых и бинарных. Согласно справке MySQL, разработанного компанией Oracle, выделяются следующие типы оптимизации структуры базы данных [8]:

1. *Оптимизация объемов данных.* Она заключается в уменьшении объемов данных, хранимых на диске для представления таблиц. Чем меньше занимает таблица в памяти, тем быстрее происходит работа с базой данных.

2. *Оптимизация типов данных.* Заключается в устранении избыточности размеров типов данных.

При разработке системы выработаны следующие правила выбора типов данных:

1. *Тип числовых данных* необходимо устанавливать исходя из предполагаемого максимально возможного содержимого данных. В случае, если есть возможность заменить иные типы на числовой без потери смысловой и структурной целостности — следует сделать это.

2. Для *строкового типа данных* необходимо всегда стараться задавать минимальный размер. Не следует все время указывать максимально возможный, в случае, если содержимое не до конца известно, лучше запросить уточнение проектных материалов.

3. *Бинарные типы данных* следует применять лишь в редких случаях, в особенности для хранения длинных строк следует тщательно изучить возможности хранения данных в строках фиксированного размера вместо использования бинарных типов.

4. Выводы

Для достижения максимальной производительности выполнения запросов необходимо искать более эффективные решения, анализируя процесс выполнения запросов сервером. К примеру, по возможности заменять запросы с выборкой без повторений (с использованием ключевого слова «distinct») запросами с использованием включения «exists». Также использовать индексы для тех полей, по которым идет частый отбор, например, для ключевых полей.

Литература

[1] *Bentley J.* Writing Efficient Programs. — Prentice Hall Ptr, 1982

[2] *Pluzhnik E. V., Nikulchev E. V.* Use of dynamical systems modeling to hybrid cloud database // International Journal of Communications, Network and System Sciences. 2013. Vol. 6. No. 12. С. 505–512.

[3] *Плужник Е. В., Никульчев Е. В.* Слабоструктурированные базы данных в гибридной облачной инфраструктуре // Современные проблемы науки и образования. 2013. № 4. С. 95.

[4] Введение в базы данных [Электронный ресурс] . — Режим доступа: http://www.codenet.ru/progr/vbasic/vb_db/1.php

[5] *Кириллов В. В.* Основы проектирования реляционных баз данных. Учебное пособие. — СПб. : ИТМО, 1994.

[6] *Matthias Jarke, Jurgen Koch.* Query Optimization in Database Systems // Computing Surveys, Vol. 16, No. 2, 1984.

[7] *Israel M., Jones J. S., Jones S.* MCSE: SQL Server 2000 Design Study Guide (Exam 70-229) . — Sybex, 2001

[8] Optimizing Database Structure [Электронный ресурс] . — Режим доступа: http://dev.mysql.com/doc/refman/5.5/en/optimizing-database-structure.html.

Автор:
Федосеенко Татьяна Александровна, магистрант Московского технологического института

Using Methods of Query Optimizatio to the Distributed Database to Efficiency Improve of Information Systems

T. A. Fedoseenko
Moscow Technological Institute
38a Leninsky Pr., Moscow, 119334,
e-mail: velliar@gmail.com

Abstract. Article is devoted to the task of improving the efficiency of distributed databases. To increase query performance solutions must be sought by analyzing the process of querying the server: replace queries with no sample in repetition, use indexes to optimize the structure of the database at the stage of physical design.
Keywords: database, query optimization, distributed databases.

Reference

[1] *Bentley J.* (1982) Writing Efficient Programs. Prentice Hall Ptr.

[2] *Pluzhnik E. V.*, *Nikulchev E. V.* (2013) Use of dynamical systems modeling to hybrid cloud database. *International Journal of Communications, Network and System Sciences*, 6(12), 505–512.

[3] *Pluzhnik E. V.*, *Nikulchev E. V.* (2013) Slabostrukturirovannye bazy dannyh v gibridnoj oblachnoj infrastructure. *Sovremennye problemy nauki i obrazovanija*, 4, 95. (In rus.)

[4] http://www.codenet.ru/progr/vbasic/vb_db/1.php

[5] *Kirillov V. V.* (1994) Osnovy proektirovanija reljacionnyh baz dannyh. Uchebnoe posobie. SPbITMO. (In rus.)

[6] *Matthias Jarke, Jurgen Koch.* (1984) Query Optimization in Database Systems. *Computing Surveys*, 16(2).

[7] *Israel M.*, *Jones J. S.*, *Jones S.* (2001) MCSE: SQL Server 2000 Design Study Guide (Exam 70-229) . Sybex,

[8] Optimizing Database Structure http://dev.mysql.com/doc/refman/5.5/en/optimizing-database-structure.html.