Интернет-журнал «Науковедение» ISSN 2223-5167 http://naukovedenie.ru/

Том 8, №3 (2016) http://naukovedenie.ru/index.php?p=vol8-3

URL статьи: http://naukovedenie.ru/PDF/81TVN316.pdf

Статья опубликована 22.06.2016.

Ссылка для цитирования этой статьи:
Петрова А.Н., Калмыков И.Н. Исследование рекомендаций по устранению причин неэффективности запросов к таблицам базы данных // Интернет-журнал «НАУКОВЕДЕНИЕ» Том 8, №3 (2016) http://naukovedenie.ru/PDF/81TVN316.pdf (доступ свободный). Загл. с экрана. Яз. рус., англ.

УДК 004

Петрова Анна Николаевна
ФГБОУ ВО «Комсомольский-на-Амуре государственный технический университет», Россия, Комсомольск-на-Амуре[1]
Доцент кафедры «Математическое обеспечение и применение ЭВМ»
Кандидат технических наук
E-mail: PetrovaAN2006@yandex.ru
РИНЦ: http://elibrary.ru/author_profile.asp?id=650995

Калмыков Илья Никифорович
ФГБОУ ВО «Комсомольский-на-Амуре государственный технический университет», Россия, Комсомольск-на-Амуре
Магистр
E-mail: kalmykoff2020@icloud.com

Исследование рекомендаций по устранению причин неэффективности запросов к таблицам базы данных

Аннотация. Одним из методов повышения производительности работы приложений, связанных с базой данных, является оптимизация запросов к таблицам базы данных. Важность вопроса оптимизации запроса повышается с ростом количества записей до десятков и сотен тысяч в таблицах, к которым обращается запрос. Существуют различные рекомендации по устранению причин замедления выполнения запросов. Однако в научной литературе не приводятся результаты экспериментальных исследований, по эффективности применения тех или иных рекомендаций. Авторами были проведены численные эксперименты по исследованию как существующих, так и предложенных авторами рекомендаций, для определения количественных значений результативности применения рекомендаций. Ряд рекомендаций иллюстрируется примерами применения рекомендаций, в том числе несколькими альтернативными вариантами. В работе приводятся отличия в эффективности применения рекомендаций в различных системах управления базами данных. В статье приводятся относительные значения эффективности применения рекомендаций сравнительно с исходными запросами. В результате исследований авторами разработан алгоритм анализа текста запроса и применения рекомендаций. Результаты исследований расширяют знания о практическом применении как известных, так и новых рекомендаций по оптимизации запросов к таблицам базы данных. Статья имеет практическую направленность и методически полезна для начинающих программистов в области языка структурированных запросов.

[1] 681013, Россия, Хабаровский край, г. Комсомольск-на-Амуре, проспект Ленина, 27, Кафедра МОП ЭВМ

Ключевые слова: базы данных; оптимизация запросов; оптимизация приложений; информационные системы

Введение

В процессе сопровождения информационной системы возникает ряд проблем [1]. Одной из которых является рос числа записей и объема баз данных, вследствие чего время выполнения запросов становится неприемлемым. Среди прочих методов ускорения работы приложений, остановимся на методах оптимизации запросов, выполняемых в системе управления базой данных (СУБД).

Общие вопросы оптимизации запросов

Оптимизация запросов реляционных баз данных состоит из двух равных по своей важности аспектов. Первый аспект не зависит от разработчика, является внутренней задачей СУБД и заключается в определении наиболее эффективного способа выполнения запроса. Второй аспект, полностью зависит от мастерства разработчика, так как заключается в написании таких запросов, для которых СУБД могла бы использовать наиболее эффективные способы нахождения данных. Особенность написания запросов на декларативном языке SQL заключается в том, что для достижения одинакового результата могут быть написаны совершенно непохожие друг на друга запросы, включающие в себя различные операторы, отличающиеся способами соединения таблиц, по-разному использующие преимущества индексации и т.д.

Существуют различные рекомендации по повышению качества написания запросов [1-10]. Однако авторы указывают при этом количественные оценки эффективности применения этих рекомендаций. Возникает ряд вопросов. По какому алгоритму программисту следует анализировать запрос для его оптимизации по времени выполнения? Для всех ли СУБД предлагаемые рекомендации работают одинаково? Ответам на поставленные вопросы посвящена данная статья.

Одним из ключевых понятий при оптимизации запросов является *план запроса*. План запроса – это методы и порядок их применения, с помощью которых СУБД может получить желаемый результат. После лексического и синтаксического анализа текста запроса происходит его оптимизация, после чего, СУБД выбирает наиболее подходящий план, выполняет его.

Необходимо отметить, что в вопросе организации индексов, у разработчиков различных СУБД имеются свои подходы. Для СУБД MS SQL Server существует два вида индекса: кластеризованные (кластерные) и некластеризованные (некластерные). Данные в таблице хранятся в отсортированном виде только в случае если создан кластеризованный индекс у этой таблицы. Таблица, не имеющая кластеризованного индекса, называется кучей. СУБД типа Firebird SQL или Oracle не поддерживают создание кластеризованных индексов, тем не менее поддерживают создание некластеризованных индексов.

Для определения эффективности возможных модификаций запроса были проведены тесты на базе данных предприятия Комсомольский Филиал Акционерного общества "Газпром газораспределение Дальний Восток". База данных, подлежащая анализу, ведется с 2000 года, поддерживается системой управления базами данных Firebird SQL. Объем хранимых данных превышает десяток гигабайт, количество таблиц – несколько сотен. За 16 лет существования база данных накопила в себе миллионы записей, поэтому вопрос организации быстрого доступа к данным и прозрачной логики SQL-инструкций очень актуален.

Оригинальная база данных предприятия поддерживается средствами Firebird SQL, но для сравнительного анализа некоторых рекомендаций структура оригинальной базы данных была воссоздана в среде MS SQL Server.

Причины неэффективности запроса и рекомендации для их устранения

Рассмотрим причины неэффективности запроса, рекомендации по их устранению и их эффективность:

1. Н*е используются синонимы для таблиц*.

Синонимы для таблиц следует применять во всех запросах, в которых упоминается больше одной таблицы в предложении FROM. Использование синонимов в таких запросах хоть и незначительно ускоряет операцию разбора SQL оператора ядром СУБД, за счет уменьшения рекурсии запросов, тем не менее повышает читаемость, упрощает разбор код разработчиком. Эффективность данной рекомендации составляет 1,42%.

2. *Неправильный порядок перечисления условий в секции FROM.*

Порядок перечисления таблиц в секции FROM имеет значение для оптимизатора, когда он выбирает способ выполнения запроса. Оптимизатор работает следующим образом: сначала он просматривает предложения WHERE и присваивает таблицам определенный вес, основываясь на типе предиката, затем он выбирает таблицу с наименьшей стоимостью и делает ее управляющей таблицей. Однако, есть один тонкий момент, который заключается в том, что если несколько таблиц получают одинаковую стоимость, и она является наименьшей, то оптимизатор выбирает в качестве управляющей ту таблицу, которая стоит последней в предложении FROM. Поэтому последней в списке должна быть таблица, возвращающая наименьшее количество строк.

В ходе эксперимента были выбран запрос, в котором две таблицы, содержащие различное количество записей, согласно выводам оптимизатора СУБД, получили одинаковую стоимость выполнения. Для выяснения истинности данной рекомендации запрос был выполнен в двух вариациях: одна из двух, одинаковых по стоимости, таблиц, возвращающая наименьшее количество записей была указана перед таблицей, возвращающей наибольшее количество записей и наоборот. Запрос, в котором таблица возвращала наименьшее количество строк стояла последней выполнился на 932 мс быстрее, эффективность данной рекомендации составила 9,89%.

Таким образом, после формирования предварительного плана запроса необходимо посмотреть стоимости таблиц, участвующих в запросе. В случае, если для нескольких таблиц стоимость будет одинакова и запрос требует ускорения выполнения, таблицу, возвращающую наименьшее количество записей необходимо указать в конце секции FROM.

3. *Запрос, содержащий операторы EXIST (IN) и DISTINCT периодически работает медленно*.

Эффективность операторов EXISTS и IN зависит от количества строк, возвращаемых каждым из подзапросов.

В запросе с использованием IN управляющей таблицей является подзапрос, указанный в предложении IN, основной запрос повторяется для каждой строки возвращаемой подзапросом в IN. В запросе с использованием EXISTS, наоборот, управляющим является основной запрос, и подзапрос, указанный в EXISTS повторяется для каждой строки, выбираемой в основном запросе.

Таким образом, если подзапрос возвращает малое количество строк, а основной запрос возвращает большое количество, то следует использовать оператор IN, иначе необходимо использовать оператор EXISTS.

Однако, существуют частные случаи, когда необходимо найти первую встречную строку по значению некоторого поля и аналогичного результата можно добиться и с помощью оператора DISTINCT. В базе данных типа Firebird SQL либо Oracle оператор DISTINCT заставит ядро СУБД просматривать поочередно все записи из искомой таблицы, в базе данных, организованной посредством MS SQL Server, будет произведен просмотр кластеризованного индекса, если он имеется, либо полный просмотр таблицы, если индекса не существует, пока не будет встречено искомое значение. В случае, если искомое значение находится в первых строках таблицы, скорость нахождения строки будет крайне высока, но в случае, когда искомая строка находится ближе к концу таблицы, поиск будет затруднителен. И в данном случае применение оператора EXIST или IN будет более эффективным, нежели применение оператора DISTINCT.

Рекомендация: запрос необходимо разбить на два подзапроса, используя оператор EXIST или IN. В ходе эксперимента эффективность данной рекомендации составила 95,04%.

4. *Излишнее употребление операторов операций над множествами строк.*

Для применения к множествам строк булевых операций (вычитание, объединение, дополнение и пр.) необходимо четко представлять структуру базы и, при использовании подзапросов, необходимо иметь представление о приблизительном количестве записей, возвращаемых каждым подзапросов. В большинстве случаев, при грамотном использовании операций над множествами строк, можно повысить значительно эффективность. Для исследуемой базы данных, эффективность исключения одного оператора EXIST, а также пересмотр отношений множеств строк, и модификация текста запроса привели к увеличению эффективности на 97%.

5. *Сочетание согласованных изменений таблицы с отсутствием необходимых индексов.*

Одна из наиболее медленных команд в SQL команда UPDATE, поскольку большинство согласованных изменений в таблицах требуют полного просмотра таблиц. В результате эти операции являются ресурсоемкими и очень медленными, когда таблицы слишком большие. Проблема низкой производительности возникает потому, что в запросе не используется информация из таблицы для ограничения количества просматриваемых строк. Если какое-либо условие в предложении WHERE оператора UPDATE не отсечет большинство строк, то оператор UPDATE потребует огромного количества процессорного времени.

Способы отсечения лишних строк могут быть различными. Рассмотрим макет (листинг 1) типового запрос UPDATE. Количество записей в таблице ABONENT1 существенно больше, чем количество записей в таблице ABONENT2.

Листинг 1 - Оригинальный запрос UPDATE

```
UPDATE ABONENT1
    SET VAL1 = (SELECT F1
                FROM ABONENT2
                WHERE ABONENT1.ID = ABONENT2.FID
                )
```

При выполнении данного запроса ядро СУБД будет вынуждено просмотреть все записи из таблицы ABONENT1, и, в случае, если ID абонентов из обеих таблиц будет совпадать, то поле VAL1 будет обновлено. Поскольку количество записей из таблицы ABONENT1, будут подвергаться попытке обновления даже те, которые заведомо не удовлетворяют условию, то это повлечет за собой медленное выполнение запроса. Поэтому возможны модификации исходного запроса, при котором лишние строки будут отсечены.

Первый вариант модификации запроса (листинг 2) с использованием IN, эффективность которой составила 24,11%.

Листинг 2 - Модификация UPDATE с помощью IN

```
UPDATE ABONENT1
    SET VAL1 = (SELECT F1
                FROM ABONENT2 A2
                WHERE A2.FID IN (SELECT A3.ID
                                 FROM ABONENT2 A3
                                 WHERE A3.FID = ABONENT1.ID
                                 )
               )
```

Однако, данный запрос можно модифицировать и с применением оператора EXIST (листинг 3). Преимущество такой модификации будет заключаться в нахождении оператором EXIST соответствия между малым количеством записей из таблицы ABONENT2 и большим из таблицы ABONENT1.

Листинг 3 - Модификация UPDATE с помощью EXIST

```
UPDATE ABONENT1
    SET VAL1 = (SELECT 1
                FROM ABONENT2 A3
                WHERE A2.FID = A1.ID
                )
    WHERE (EXISTS SELECT *
                  FROM ABONENT2 A2
                  WHERE A2.FID = ABONENT1.ID
          )
```

Важным условием применения данной рекомендации является наличие индекса по специальному полю из подзапросов. В данном случае высокая эффективность обеспечивается за счет сочетания операций над множеством строк и наличия индекса по полю ID, что позволяет не сканировать таблицы с наибольшим количеством записей целиком. Результаты проверки приведены в таблице 1, составленной авторами.

Таблица 1

Сравнение результатов выполнения вариантов запросов

Варианты запросов	Время выполнения, (мс)	Процент эффективности относительно листинг 1, (%)	Процент эффективности относительно листинг 2, (%)
Листинг 1 (=)	1750		
Листинг 2 (IN)	1328	24,11	
Листинг 3 (EXIST)	140	92,46	89,46

Столь значимое предпочтение варианта с EXIST (Листинг 3) над IN (Листинг 2) объясняется соотношением количества записей в таблицах A1 - более 100 тысяч записей, в A2 - 8 записей.

Рекомендация: по возможности заменить сравнение на IN или EXIST, в зависимости от соотношения числа записей в таблицах.

6. *Отсутствие необходимого индекса при выполнении не соотнесенного запроса.*

На листинге 4 представлен макет запроса, выполняющего не соотнесенные изменения.

Листинг 4 – Пример не соотнесенного запроса

```
SELECT A1.ID
FROM ABONENT1 A1
WHERE A1.ID IN (SELECT A2.FID
                FROM ABONENT2 A2
                )
```

Как известно, ядро СУДБ не использует индекс на поле, если в секции WHERE на него не налагается некое условие. Частным случаем не соблюдения данного правила является пример не соотнесенного запроса, представленного на листинге 4. У подзапроса отсутствует секция WHERE, тем не менее в данном случае ядро СУБД будет пытаться использовать индекс для поля A2.FID в том случае, если он имеется. Задача разработчика, установить индекс на поле A2.FID. Эффективность данной рекомендации составляет 47,17%. Можно сказать, что эффективность данной рекомендации будет зависеть от того, насколько быстрее будет выполняться индексное сканирование таблицы A2 в отличие от ее полного сканирования.

7. *Непреднамеренный запрет индексов.*

Для СУБД MS Firebird SQL, Oracle существуют различные способы непреднамеренной блокировки индексов, некоторые из них представлены в таблице 2, составленной авторами.

Таблица 2

Примеры условий поиска

Условия поиска, приводящие к блокировке индексного поиска	Условия поиска, не блокирующие индекс для поиска	Эффективность 2 относительно 1 в Firebird SQL
1	2	3
ID = ‘992000000’	ID = 992000000	(15-0)/15=100%
E <> 3	E <3 OR E> 3	(31-15)/31=51,61%
E NOT IN ('2', '32')	E<2 OR (E>2 AND E<32) OR E>32	(33-15)/33=54,55%
P * 2 <1000	P < 500	(500-16)/500=96,80%

Правильное употребление аргумента поиска (условия в секции WHERE) критично для СУБД Firebird SQL, чего нельзя сказать о MS SQL Server, в которой оптимизатор выберет индекс (при его наличии) согласно его стоимости. Если стоимость некластеризованного индекса окажется меньше стоимости кластеризованного, то, даже при неправильном употреблении аргумента поиска, запрос выполнится за приемлемое время, так как будет произведен поиск по кластеризованному индексу, другими словами, в СУБД MS SQL Server невозможно заблокировать индексный поиск, при наличии кластеризованного индекса.

Рекомендация: переформулировать условия, блокирующие индексный поиск, если используется СУДБ MS Firebird SQL, Oracle, для СУБД MS SQL Server данная рекомендация не имеет значения.

8. *Правило 10,15, 20% не соблюдается.*

В ходе экспериментов было выявлено, что использование индексов в запросах оправдано, если запрос извлекает меньше 15% (10, 20) строк из таблицы. Во всех остальных случаях полный просмотр таблицы будет работать быстрее. Это правило справедливо и для MS SQL Server и для Firebird SQL.

Заключение

В качестве выводов можно сказать, следующее:

1. Существуют рекомендации, из представленных выше, которые не приносят ощутимого выигрыша во времени, но тем не менее настоятельно рекомендованы к использованию. К таким можно отнести, например, рекомендацию с использованием синонимов, рекомендации ориентировочного вычисления количества записей из каждой таблицы, участвующей в запросе.
2. Для применения некоторых рекомендаций стоит обращать внимание на структуру запроса, в частности – используются ли подзапросы, применяются ли операторы операций над множествами строк типа EXIST, IN, UNION и прочие. В случае обнаружения подобных свойств, необходимо оценить эффективность обработки каждого из множества, возможно, прибегнув к операциям над множествами типа объединения, вычитания, дополнения и др., либо как минимум иметь представление о количестве записей, удовлетворяющих условиям запроса.
3. Некоторые запросы могут быть частным случаем, например, так называемые «не соотнесенные» запросы, и требовать грамотного индексирования полей таблиц, участвующих в запросе.

4. Необходимо аккуратно относиться к использованию оператора DISTINCT, так как его применение не всегда выгодно ввиду зависящего от позиции искомой строки в таблице.
5. СУБД MS SQL Server и Firebird SQL показали различные подходы к организации индексов в своих системах.

Таким образом, в результате исследований можно предложить следующий **порядок анализа запроса с целью его оптимизации**:

1. Проверить наличие псевдонимов. Если в запросе участвуют более одной таблицы, то следует использовать псевдонимы.
2. Проверить условия поиска. Если СУБД не MS SQL Server, и планируется использование индексов, то должны применяться не блокирующие условия поиска.
3. Проверить наличие предложения DISTINCT, если есть, то проанализировать как далеко от начала таблицы расположена искомая запись. Если во второй трети и далее, то изменить конструкцию запроса, например, использовать IN или EXIST.
4. Если в предложении FROM использовано несколько источников, то оценить, по возможности, количество записей, возвращаемых каждым из них и поставить условия по убыванию (убывать должно количество удовлетворяемых условиям записей).
5. Проверить наличие предложения IN или EXIST проанализировать количество записей, удовлетворяющих условиям отбора.
6. Проверить наличие операторов реляционной алгебры UNION, INTERSECT и т.д., если есть, то проанализировать, можно ли обойтись без данных операторов. В процессе проектирования запроса можно попробовать использовать только секцию WHERE и подзапрос в ней, либо использовать операторы реляционной алгебры совместно с подзапросами. Выбрать оптимальный вариант.
7. Настроить обслуживание базы данных так, чтобы регулярно выполнялись обновления индексов и статистики.

ЛИТЕРАТУРА

1. Лошманов, А.Ю. Организация работ по сопровождению информационной системы ВУЗа / А.Ю. Лошманов, Я.Ю. Григорьев, А.Н. Петрова // Интернет-журнал «Науковедение», 2013 №4 (17) [Электронный ресурс]-М.: Науковедение, 2013. - Режим доступа: http://naukovedenie.ru/PDF/66tvn413.pdf, свободный. – Загл. с экрана. - Яз. рус., англ.
2. Евсеев, Г.С. Исследование влияния уровня нормализации таблиц базы данных на время выполнения запросов / Г.С. Евсеев, Д.М. Ильинская // Сборник трудов конференции "Научная сессия ГУАП". Санкт-Петербург: ГУАП, 2015. С.193-196.
3. Михеевич, В. Причины неэффективности SQL-запросов в Oracle. Оптимизация производительности SQL-запросов / В. Михеевич // Системный администратор. 2015. №6. С. 47-51.
4. Борри, Х. Firebird: руководство разработчика баз данных / Х. Борри. - Спб.: БХВ-Петербург, 2006. 1104 с
5. Кузнецов С.Д. Оценка эффективности минимизации ограничений запросов к субд / С.Д. Кузнецов, Н.А. Мендкович // Труды института системного программирования РАН №25 с. 113-130.
6. Дейт, К.Дж. Введение в системы баз данных: пер. с англ. / К.Дж. Дейт. - М.: Издательский дом "Вильямс", 2002. 1072 с.
7. Маркин, А.В. Построение запросов и программирование на SQL / А.В. Маркин. - М.: Диалог-МИФИ, 2008. 318 с.
8. Блудов, И.В. Особенности табличных выражений SQL и их соответствие с концепциями реляционной модели данных / И.В. Блудов. // Труды института системного программирования РАН. 2013. №24. С. 417-436.
9. Атрощенко, В.А. Базы данных / В.А. Атрощенко, Р.А. Дьяченко, Н.Д. Чигликова, В.Е. Бельченко, Е.С. Белодед, И.С. Лоба. - Армавир:Армавирская государственная педагогическая академия, 2015. 80 с.
10. Малков, О.Б. Работа с Transact-SQL / О.Б. Малков, М.В. Девятерикова. - Омск: Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования "Омский государственный технический университет", 2015. 136 с.

Petrova Anna Nikolaevna
Komsomolsk-na-Amure state technical university, Russia, Komsomolsk-na-Amure
E-mail: PetrovaAN2006@yandex.ru

Kalmykov Ilya Nikiforovich
Komsomolsk-na-Amure state technical university, Russia, Komsomolsk-na-Amure
E-mail: kalmykoff2020@icloud.com

The study of recommendations for elimination of inefficiency queries causes to database tables

Abstract. One method of increasing the productivity of applications associated with a database is to optimize the query to the database tables. The importance of the optimization issue of the query increases with the number of entries up to tens and hundreds of thousands in the tables accessed by the query. There are various recommendations to address the causes of query execution slowing down. However, the scientific literature does not present the results of experimental studies on the effectiveness of those or other recommendations. The authors have performed numerical experiments to investigate both existing and proposed by the author recommendations for quantifying the impact of the application of the values of recommendations. The number of recommendations is illustrated by examples of the recommendations application, including several alternatives. The paper presents the differences in the efficacy of the recommendations in a variety of database management systems. The article presents the relative values of the recommendations effectiveness as compared with the original request. As a result of study authors developed an algorithm for the query text analysis and application of the recommendations. The results of the research expand the knowledge on the practical application of both known and new recommendations for optimizing queries to database tables. The article has a practical focus and methodically useful for beginners in the field of structured query language.

Keywords: database; query optimization; application optimization; information systems

REFERENCES

1. Loshmanov, A.Yu. Organizatsiya rabot po soprovozhdeniyu informatsionnoy sistemy VUZa / A.Yu. Loshmanov, Ya.Yu. Grigor'ev, A.N. Petrova // Internet-zhurnal «Naukovedenie», 2013 №4 (17) [Elektronnyy resurs]-M.: Naukovedenie, 2013. - Rezhim dostupa: http://naukovedenie.ru/PDF/66tvn413.pdf, svobodnyy. – Zagl. s ekrana. - Yaz. rus., angl.
2. Evseev, G.S. Issledovanie vliyaniya urovnya normalizatsii tablits bazy dannykh na vremya vypolneniya zaprosov / G.S. Evseev, D.M. Il'inskaya // Sbornik trudov konferentsii "Nauchnaya sessiya GUAP". Sankt-Peterburg: GUAP, 2015. S.193-196.
3. Mikheevich, V. Prichiny neeffektivnosti SQL-zaprosov v Oracle. Optimizatsiya proizvoditel'nosti SQL-zaprosov / V. Mikheevich // Sistemnyy administrator. 2015. №6. S. 47-51.
4. Borri, Kh. Firebird: rukovodstvo razrabotchika baz dannykh / Kh. Borri. - Spb.: BKhV-Peterburg, 2006. 1104 s
5. Kuznetsov S.D. Otsenka effektivnosti minimizatsii ogranicheniy zaprosov k subd / S.D. Kuznetsov, N.A. Mendkovich // Trudy instituta sistemnogo programmirovaniya RAN №25 s. 113-130.
6. Deyt, K.Dzh. Vvedenie v sistemy baz dannykh: per. s angl. / K.Dzh. Deyt. - M.: Izdatel'skiy dom "Vil'yams", 2002. 1072 s.
7. Markin, A.V. Postroenie zaprosov i programmirovanie na SQL / A.V. Markin. - M.: Dialog-MIFI, 2008. 318 s.
8. Bludov, I.V. Osobennosti tablichnykh vyrazheniy SQL i ikh sootvetstvie s kontseptsiyami relyatsionnoy modeli dannykh / I.V. Bludov. // Trudy instituta sistemnogo programmirovaniya RAN. 2013. №24. S. 417-436.
9. Atroshchenko, V.A. Bazy dannykh / V.A. Atroshchenko, R.A. D'yachenko, N.D. Chiglikova, V.E. Bel'chenko, E.S. Beloded, I.S. Loba. - Armavir:Armavirskaya gosudarstvennaya pedagogicheskaya akadeniya, 2015. 80 s.
10. Malkov, O.B. Rabota s Transact-SQL / O.B. Malkov, M.V. Devyaterikova. - Omsk: Federal'noe gosudarstvennoe byudzhetnoe obrazovatel'noe uchrezhdenie vysshego professional'nogo obrazovaniya "Omskiy gosudarstvennyy tekhnicheskiy universitet", 2015. 136 s.