|  |  |  |
| --- | --- | --- |
| INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 4, ISSUE 10, OCTOBER 2015 ISSN 2277-8616 | | |
| Query Optimization Techniques - Tips For Writing Efficient And Faster SQL Queries  Методы Оптимизации Запросов - Советы По Написанию Эффективные И Быстрые SQL-Запросы | | |
|  | Jean HABIMANA |  |
| Abstract: SQL statements can be used to retrieve data from any database. If you've worked with databases for any amount of time retrieving information, it's practically given that you've run into slow running queries. Sometimes the reason for the slow response time is due to the load on the system, and other times it is because the query is not written to perform as efficiently as possible which is the much more common reason. For better performance we need to use best, faster and efficient queries. This paper covers how these SQL queries can be optimized for better performance. Query optimization subject is very wide but we will try to cover the most important points. In this paper I am not focusing on, in- depth analysis of database but simple query tuning tips & tricks which can be applied to gain immediate performance gain | | |
| Аннотация: Операторы SQL могут использоваться для извлечения данных из любой базы данных. Если вы работали с базами данных в течение какого-либо периода времени, извлекая  информацию, практически наверняка вы сталкивались с медленно выполняемыми запросами. Иногда причиной медленного времени отклика является нагрузка на систему, а в других случаях это связано с тем, что запрос написан не для максимально эффективного выполнения, что является гораздо более распространенной причиной. Для повышения производительности нам необходимо использовать лучшие, более быстрые и эффективные запросы. В этой статье рассматривается, как эти SQL-запросы могут быть оптимизированы для повышения производительности. Тема оптимизации запросов очень широка, но мы постараемся охватить наиболее важные моменты. В этой статье я сосредотачиваюсь не на углубленном анализе  базы данных, а на простых советах и хитростях по настройке запросов, которые можно применить для немедленного повышения производительности | | |
|  | 1. INTRODUCTION   Query optimization is an important skill for SQL developers and database administrators (DBAs). In order to improve the performance of SQL queries, developers and DBAs need to understand the query optimizer and the techniques it uses to select an access path and prepare a query execution plan. Query tuning involves knowledge of techniques such as cost-based and heuristic-based optimizers, plus the tools an SQL platform provides for explaining a query execution plan.The best way to tune performance is to try to write your queries in a number of different ways and compare their reads and execution plans. In this paper I proposed various techniques that you can use to try to optimize your database queries. | I. ВВЕДЕНИЕ  Оптимизация запросов является важным навыком для разработчиков SQL и администраторов баз данных (СУБД).  Чтобы повысить производительность SQL-запросов, разработчикам и администраторам баз данных необходимо понимать оптимизатор запросов и методы, которые он использует для выбора пути доступа и подготовки плана выполнения запроса. Настройка запросов  требует знания таких методов, как оптимизаторы на основе затрат и эвристики, а также инструментов , предоставляемых платформой SQL для объяснения плана выполнения запроса. Лучший способ настроить производительность - попытаться записать свои запросы в  количество различных способов и сравните их чтения и планы выполнения. В этой статье я предложил различные методы, которые вы можете использовать, чтобы попытаться оптимизировать свои запросы к базе данных. |
|  | II. GENERAL TIPS FOR QUERY OPTIMIZATION Each tip was tested by running both the original query and improved query while retrieving information from the Oracle 11g sample database especially on Sales schema. I recorded the average time of each query to show the speed increase of using the more efficient query | II. ОБЩИЕ СОВЕТЫ ПО ЗАПРОСУ  Оптимизация Каждый совет был протестирован путем выполнения как исходного запроса, так и  улучшенного запроса при извлечении информации из Oracle 11g пример базы данных, особенно по схеме продаж. Я записал среднее время каждого запроса, чтобы показать увеличение скорости использования более эффективного запроса |
|  | Tip #1: Use Column Names Instead of \* in a SELECT Statement If you are selecting only a few columns from a table there is no need to use SELECT \*. Though this is easier to write, it will cost more time for the database to complete the query. By selecting only the columns you need, you are reducing the size of the result table, reducing the network traffic and also in turn boosting the overall performance of the query. | Совет №1:  Используйте имена столбцов вместо \* в SELECT  Совет: Если вы выбираете только несколько столбцов из таблицы  , нет необходимости использовать SELECT \*.  Хотя это проще в написании, для выполнения запроса базе данных потребуется больше времени.  Выбирая только нужные вам столбцы, вы уменьшаете размер результирующей таблицы, сокращаете сетевой трафик и , в свою очередь, повышаете общую производительность запроса. |
|  | Tip #2:  Avoid including a HAVING clause in SELECT  statements  The HAVING clause is used to filter the rows after all the  rows are selected and it is used like a filter. It is quite  useless in a SELECT statement. It works by going through  the final result table of the query parsing out the rows that  don’t meet the HAVING condition. | Совет №2:  Избегайте включения оператора HAVING в SELECT  Оператор HAVING используется для фильтрации строк после того, как  агрегация произведена, и оно используется как фильтр.  Когда WHERE используется до агрегации. |
|  | Tip #3:  Eliminate Unnecessary DISTINCT Conditions  Considering the case of the following example, the  DISTINCT keyword in the original query is unnecessary  because the table\_name contains the primary key p.ID,  which is part of the result set. | Совет №3:  Устраните ненужные ОТДЕЛЬНЫЕ условия  Рассматривая случай следующего примера, ключевое слово DISTINCT в исходном запросе не требуется  , поскольку имя\_таблицы содержит первичный ключ p.ID ,  который является частью результирующего набора.  Example:  Original query:  SELECT DISTINCT \* FROM SH.sales s  JOIN SH.customers c  ON s.cust\_id= c.cust\_id  WHERE c.cust\_marital\_status = 'single';  Improved query:  SELECT \* FROM SH.sales s JOIN  SH.customers c  ON s.cust\_id = c.cust\_id  WHERE c.cust\_marital\_status='single'; |
|  | Tip #4:  Un-nest sub queries  Rewriting nested queries as joins often leads to more  efficient execution and more effective optimization. In  general, sub-query un-nesting is always done for correlated  sub-queries with, at most, one table in the FROM clause,  which are used in ANY, ALL, and EXISTS predicates. A  uncorrelated sub-query, or a sub-query with more than one  table in the FROM clause, is flattened if it can be decided,  based on the query semantics, that the sub-query returns at  most one row | Совет №4:  Извлекать вложенные запросы из гнезда Переписывание вложенных запросов в виде объединений часто приводит к более эффективному выполнению и более действенной оптимизации. В общем, отмена запроса всегда выполняется для коррелированных подзапросов, содержащих не более одной таблицы в предложении FROM, которые используются в предикатах ANY, ALL и EXISTS. Некоррелированный подзапрос или подзапрос с более чем одной  таблицей в предложении FROM сглаживается, если на основе семантики запроса можно решить, что подзапрос возвращает неболее одной строки  Original query:  SELECT \*  FROM SH.products p  WHERE p.prod\_id =  (SELECT s.prod\_id  FROM SH.sales s  WHERE s.cust\_id = 100996  AND s.quantity\_sold = 1 );  Improved query:  SELECT p.\*  FROM SH.products p, sales s  WHERE p.prod\_id = s.prod\_id  AND s.cust\_id = 100996  AND s.quantity\_sold = 1; |
|  | Tip #5:  Consider using an IN predicate when querying an  indexed column  The IN-list predicate can be exploited for indexed retrieval  and also, the optimizer can sort the IN-list to match the sort  sequence of the index, leading to more efficient retrieval.  Note that the IN-list must contain only constants, or values  that are constant during one execution of the query block,  such as outer references. | Совет №5:  Рассмотрите возможность использования предиката IN при запросе индексированного столбца  Предикат IN-list может быть использован для индексированного поиска , а также оптимизатор может отсортировать IN-list в соответствии  с последовательностью сортировки индекса, что приводит к более эффективному поиску.  Обратите внимание, что ВХОДЯЩИЙ список должен содержать только константы или значения, которые являются постоянными во время одного выполнения блока запроса,такие как внешние ссылки.  Original query:  SELECT s.\*  FROM SH.sales s  WHERE s.prod\_id = 14  OR s.prod\_id = 17;  Improved query:  SELECT s.\*  FROM SH.sales s  WHERE s.prod\_id IN (14, 17); |
|  | Tip #6:  Use EXISTS instead of DISTINCT when using table  joins that involves tables having one-to-many  relationships  The DISTINCT keyword works by selecting all the columns  in the table then parses out any duplicates.Instead, if you  use sub query with the EXISTS keyword, you can avoid  having to return an entire table. | Совет №6:  Используйте EXISTS вместо DISTINCT при использовании таблицы.  соединения, включающие таблицы, имеющие связь один-ко-многим  отношения  Ключевое слово DISTINCT работает, выбирая все столбцы.  в таблице, затем анализирует все дубликаты. Вместо этого, если вы  используйте подзапрос с ключевым словом EXISTS, вы можете избежать  необходимость вернуть всю таблицу.  Original query:  SELECT DISTINCT c.country\_id, c.country\_name  FROM SH.countries c,SH.customers e  WHERE e.country\_id = c.country\_id;  Improved query:  SELECT c.country\_id, c.country\_name  FROM SH.countries c  WHERE EXISTS (SELECT 'X' FROM SH.customers e  WHERE e.country\_id = c.country\_id); |
|  | Tip #7:  Try to use UNION ALL in place of UNION  The UNION ALL statement is faster than UNION, because  UNION ALL statement does not consider duplicate s, and  UNION statement does look for duplicates in a table while  selection of rows, whether or not they exist. | Совет №7:  Попробуйте использовать UNION ALL вместо UNION.  Оператор UNION ALL выполняется быстрее, чем UNION, потому что  Оператор UNION ALL не учитывает дубликаты и  Оператор UNION ищет дубликаты в таблице, а выбор строк, независимо от того, существуют они или нет.  Example:  Original query:  SELECT cust\_id  FROM SH.sales  UNION  SELECT cust\_id  FROM customers;  Improved query:  SELECT cust\_id  FROM SH.sales  UNION ALL  SELECT cust\_id  FROM customers; |
|  | Tip #8:  Avoid using OR in join conditions  Any time you place an ‘OR’ in the join condition, the query  will slow down by at least a factor of two. | Совет №8:  Избегайте использования OR в условиях соединения.  Каждый раз, когда вы помещаете «ИЛИ» в условие соединения, запрос  замедлится как минимум в два раза.  Example:  Original query:  SELECT \*  FROM SH.costs c  INNER JOIN SH.products p ON c.unit\_price  p.prod\_min\_price OR c.unit\_price = p.prod\_list\_price;  Improved query:  SELECT \*  FROM SH.costs c  INNER JOIN SH.products p ON c.unit\_price  p.prod\_min\_price  UNION ALL  SELECT \*  FROM SH.costs c  INNER JOIN SH.products p ON c.unit\_price  p.prod\_list\_price; |
|  | Tip #9:  Avoid functions on the right hand side of the  operator  Functions or methods are used very often with their SQL  queries. Rewriting the query by removing aggregate  functions will increase the performance tremendously. | Совет №9:  Избегайте функций в правой части экрана.  оператор  Функции или методы очень часто используются вместе с их SQL.  запросы.  Переписывание запроса путем удаления агрегата  функции значительно повысят производительность.  Example:  Original query:  SELECT \*  FROM SH.sales  WHERE EXTRACT (YEAR FROM TO\_DATE (time\_id, ‘DD-  MON-RR’)) = 2001 AND EXTRACT (MONTH FROM  TO\_DATE (time\_id, ‘DD-MON-RR’)) =12;  Improved query:  SELECT \* FROM SH.sales  WHERE  TRUNC  (time\_id)  BETWEEN  TRUNC(TO\_DATE(‘12/01/2001’,  ’mm/dd/yyyy’))  AND  TRUNC (TO\_DATE (‘12/30/2001’,’mm/dd/yyyy’)); |
|  | Tip #10:  Remove any redundant mathematics  There will be times where you will be performing  mathematics within an SQL statement. They can be a drag  on the performance if written improperly. For each time the  query finds a row it will recalculate the math. So eliminating  any unnecessary math in the statement will make it perform  faster. | Совет № 10:  Удалите всю лишнюю математику  Будут моменты, когда вы будете выступать математика в операторе SQL.  Они могут быть тормозом на исполнение, если написано неправильно.  Каждый раз, когда запрос находит строку, он пересчитывает математические значения.  Таким образом, устраняя любая ненужная математика в операторе заставит его работать Быстрее.  Example:  Original query:  SELECT \*  FROM SH.sales s  WHERE s.cust\_id + 10000 < 35000;  Improved query:  SELECT \*  FROM SH.sales s  WHERE s.cust\_id < 25000; |
|  | Query optimization is a common task performed by  database administrators and application designers in order  to tune the overall performance of the database system.  The purpose of this paper is to provide SQL scenarios to  serve as a quick and easy reference guide during the  development phase and maintenance of the database  queries. Even if you have a powerful infrastructure, the  performance can be significantly degraded by inefficient  queries. Query optimization has a very big impact on the  performance of a DBMS and it continuously evolves with  new, more sophisticated optimization strategies. So, we  should try to follow the general tips as mentioned above to  get a better performance of queries. Optimization can be  achieved with some efforts if we make it a general practice  to follow the rules. The main focus was on query  optimizations. | Оптимизация запросов — это обычная задача, выполняемая  администраторы баз данных и разработчики приложений, чтобы  для настройки общей производительности системы базы данных.  Целью данной статьи является предоставление сценариев SQL для  служить быстрым и простым справочным руководством во время  этап разработки и ведение базы данных запросы.  Даже если у вас мощная инфраструктура,  производительность может быть значительно снижена из-за неэффективности запросы.  Оптимизация запросов оказывает очень большое влияние на  производительность СУБД и она постоянно развивается вместе с  новые, более сложные стратегии оптимизации.  Итак, мы следует попытаться следовать общим советам, упомянутым выше, чтобы получить лучшую производительность запросов.  Оптимизация может быть  достигнуто с некоторыми усилиями, если мы сделаем это общей практикой  Чтобы следовать правилам.  Основное внимание уделялось запросам оптимизации. |
|  | IV. REFERENCES  [1] 10 Ways to Improve SQL Query Performance  http://www.developer.com/db/10-ways-to-improve-sql-query-performance.html  [2] 15 Ways to Optimize Your SQL Queries  http://hungred.com/useful-information/ways-optimize-sql-queries/  [3] Optimize SQL Server queries with these advanced tuning techniques.  <http://www.techrepublic.com/blog/the-enterprise-cloud/optimize-sql-server-queries-with-these-advanced-tuning-techniques/>.  [4] Making Queries Run Faster  <https://sqlschool.modeanalytics.com/advanced/faster-queries.html>.  [5] Query Optimization Techniques in Microsoft SQL Server  http://www.dbjournal.ro/archive/16/16\_4.pdf  [6] SQL Tuning or SQL Optimization. http://beginner-sql-tutorial.com/sqlquery-tuning.htm  [7] SQL Server Optimization Tips.  http://santhoshgudise.weebly.com/uploads/8/5/4/7/8547208/sql\_server\_optimization\_tips-1.doc  [8] Efficient SQL Statements  https://oracle-base.com/articles/misc/efficient-sql-statements  [9] Best Way to Write SQL Query.  http://www.ifadey.com/2010/11/best-way-to-write-sql-query/  [10] SQL Tuning Guidelines for Oracle - Simple yet Effective!  http://askanantha.blogspot.com/2007/10/sql-tuning-guidelines-for-oracle-simple.html  [11] What are the most common SQL Optimizations.  http://stackoverflow.com/questions/1332778/what-are-your-most-commonsql-optimizations  [12] Top 10 performance tuning tips for relational databases.  http://web.synametrics.com/top10performancetips.  Htm | |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |