|  |  |  |
| --- | --- | --- |
| 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. |
|  | 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 |
|  | 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. |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |