The Oracle Magician

Nov, 2004 *OracleMagician.com* *Volume III, number 4*

From the Editor

Welcome to the new issue of our online magazine, The *Oracle Magician*! This quarterly newsletter focuses on various "tricks of the trade" in the Oracle world--from DBAs, architects, developers, designers, and report writers.

Thank you to the many notes and comments from readers of my book, *The Art & Science of Oracle Performance Tuning*. Your notes are most appreciated!

In this issue we discuss when it's a good idea to use the "Not Exists" operator. By understanding the underlying principles, you can make a good choice about when to use it.

Also in this issue is a discussion of multiblock reads, and why they sometimes don't work the way we expect.

As always, we accept ideas or articles from reads that have interesting performance ideas.

Please send all ideas to Editor@OracleMagician.com

Chris Lawson

When Not Exists Should Not Exist

By Chris Lawson

Overview

I have found that there is a lot of confusion about using the operator "Not Exists." Some developers feel that it should *always* be used, while others think that it should *never* be used. Is either approach correct?

In this issue, we examine the consequences of using the ***Not Exists*** operator, and provide some practical suggestions on optimizing performance.

Let's consider a business that rents DVD movies. The manager wants to keep track of the store's inventory using their Oracle database. She has developed a query that lists the DVD titles that are not checked out at the moment.

There are two tables of interest:

DVD_Titles:	**List of all DVDs**
***Checked_Out.*:**	**DVDs rented-out**

Continued on page 2

Table of Contents

Continued from page 1

The manager's query simply fetches the titles and copy# of every single title, excluding those that are listed as checked-out. We assume we have an index on the ***Checked_Out table*** (Title). Here's the query:

SELECT Title, Copy#
FROM DVD_titles T
WHERE
Not Exists (Select 'x'
FROM Checked_Out CO
Where T.Title = CO.Title
AND T.Copy# = CO.Copy#)

Oracle must do an index scan once for *each row* in the ***DVD_titles*** table.

The execution plan for the above query is shown in **Figure 1.** In this SQL, the optimizer first performs a full scan of ***DVD_Titles***. Then, for each row, Oracle performs an index scan on the ***Checked_Out*** table to see if a matching DVD exists in that table.

So What's the Problem?

Here's the essence of the problem: In the above SQL, Oracle must do an index scan once for *each row* in the DVD_Titles table.

Since there are 100,000 rows in the ***DVD_Titles*** table, this means 100,000 index lookups. Of course, in large databases, the table might have millions of rows, making the query run time terrible.

Note that the ***Not Exists*** method does indeed eliminate a full table scan of the second table (in our case the ***Checked_Out*** table.). Unfortunately, the massive number of index scans can more than outweigh that benefit.

A Different Way

Let's take our original query and modify it to use the set operator ***MINUS***. Here's the revised SQL:

SELECT Title, Copy#
FROM Dvd_titles T
MINUS
Select Title, Copy#
FROM Checked_Out CO

Continued on page 3

Figure 1. Original Execution Plan

```
  ID PARENT OPERATION                            OBJECT_NAME
---- ------ ------------------------------------ --------------------
   0        SELECT STATEMENT
   1      0  FILTER
   2      1   TABLE ACCESS FULL                  DVD_TITLES
   3      1   TABLE ACCESS BY INDEX ROWID        CHECKED_OUT
   4      3    INDEX RANGE SCAN                  TITLE_INDEX
```

Continued from page 2

The new execution plan is shown in Figure 2. In this new plan, we eliminate 100,000 index reads by accepting a full table scan of the small ***Checked_Out*** table.

The Tradeoff

By rewriting our query using the ***MINUS*** operator, we accept a full table scan and thereby eliminate a massive number of index scans. In our example, the choice is an easy one, because the ***Checked_Out*** table is very small. Certainly scanning a small table will always be much faster than 100,000 index reads.

Also keep in mind that full table scans are very efficient. Many servers can read many blocks at a time (typically a total of 1 Mb), using multiblock reads. Index reads, however, will just read a single block at a time. This means that full table scans are often an excellent choice.

Summary:

There really isn't just one right way to design queries. In some cases, you really are better off using ***Not Exists***. In many cases, however, you should use the set operator ***MINUS***. Once you understand the principles, you can easily choose the best method for your particular case.

Updates and Notes

If you find these tuning issues interesting, I discuss performance tuning in greater detail in my book, *The Art and Science of Oracle Performance Tuning*. It is available at most large bookstores, or online at Amazon.com.

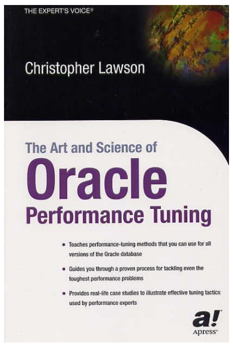
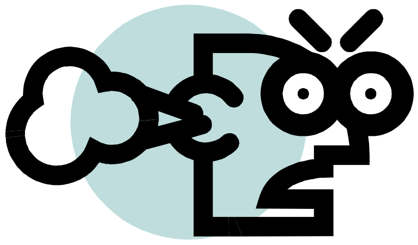

THE EXPERT'S VOICE
Christopher Lawson
The Art and Science of
Oracle
Performance Tuning
Teaches performance-tuning methods that you can use for all versions of the Oracle database
Guides you through a proven process for tackling even the toughest performance problems
Provides real-life case studies to illustrate effective tuning tactics used by performance experts
Apress

Figure 2. Revised Execution Plan

```
  ID PARENT OPERATION                           OBJECT_NAME
---- ------ ----------------------------------- -------------------
   0          SELECT STATEMENT
   1      0    MINUS
   2      1     SORT UNIQUE
   3      2      TABLE ACCESS FULL              DVD_TITLES
   4      1     SORT UNIQUE
   5      4      TABLE ACCESS FULL              CHECKED_OUT
```

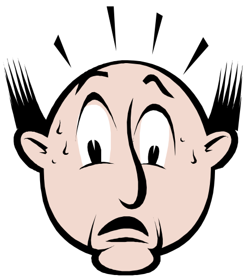
Too Few Blocks in Your Multiblock Read?



Many UNIX servers are capable of reading 1 Mb for each read. This capability is used during full table (or index) scans. It is widely known that to take advantage of your full server capability, you must set the parameter **DB_FILE_MULTIBLOCK_READ_COUNT.** However, if you're not careful, you won't get the full advantage from multiblock reads—*even if you have correctly set the parameter.*

The key point is this: the server will only read blocks that are *contiguous* on disk. By definition, an extent comprises contiguous blocks; therefore, you'll get the full multiblock read as long as you set the table (or index) extent size to 1 Mb or greater. If the extents are smaller, the server will stop the multiblock read at the end of the extent.

Tuning Traps to Avoid



In tuning queries that have lots of index scans, it's easy to underestimate the true cost of these scans. For instance, if you perform the same query several times in succession, much of the data will still be cached. This artificial situation leads to a drastically reduced runtime. In these cases, it's easy to get fooled into thinking that your query is optimized, when in reality it's your test *environment* that's been optimized.

To avoid getting fooled by data caching, try to focus on metrics that don't change so drastically from run to run. I've found that measuring logical reads (called *consistent gets* in Oracle lingo) is an excellent way to test SQL variations. The number of consistent gets will be nearly identical from run to run. When you lower these logical reads, you'll also lower disk i/o and cpu time as well.

Sizing Extents:

Steve Adams (http://www.ixora.com.au) points out that extents "should be a multiple of the multiblock read size…multiblock reads never span extent boundaries, even if the extents happen to be contiguous, and even in a locally managed tablespace. "

As Steve pointed out, however, always check for changes in recent Oracle releases. New database releases sometimes remove restrictions applicable to older versions.

The Oracle Magician

Editor: Chris Lawson
OracleMagician.com

Copyright ©2004 *The Oracle Magician*