How to use Explain Plan to optimize queries?

Asked 16 years, 2 months ago   Modified 6 years ago   Viewed 61k times

25

I have been tasked to optimize some sql queries at work. Everything I have found points to using Explain Plan to identify problem areas. The problem I can not find out exactly what explain plan is telling me. You get Cost, Cardinality, and bytes.

What do this indicate, and how should I be using this as a guide. Are low numbers better? High better? Any input would be greatly appreciated.

Or if you have a better way to go about optimizing a query, I would be interested.

sql   oracle-database   optimization

Share

Improve this question

Follow

edited Oct 25, 2008 at 8:20

Mitch Wheat
300k ● 44 ● 477 ● 550

asked Oct 24, 2008 at 17:53

Jacob Schoen
14.2k ● 15 ● 86 ● 105

4 Answers

Sorted by: Highest score (default)

11

I also assume you are using Oracle. And I also recommend that you check out the explain plan web page, for starters. There is a lot to optimization, but it can be learned.

A few tips follow:

First, when somebody tasks you to optimize, they are almost always looking for acceptable performance rather than ultimate performance. If you can reduce a query's running time from 3 minutes down to 3 seconds, don't sweat reducing it down to 2 seconds, until you are asked to.

Second, do a quick check to make sure the queries you are optimizing are logically correct. It sounds absurd, but I can't tell you the number of times I've been asked for advice on a slow running query, only to find out that it was occasionally giving wrong answers! And as it turns out, debugging the query often turned out to speed it up as well.

In particular, look for the phrase "Cartesian Join" in the explain plan. If you see it there, the chances are awfully good that you've found an unintentional cartesian join. The usual pattern for an unintentional cartesian join is that the FROM clause lists tables separated by comma, and the join conditions are in the WHERE clause. Except that one of the join conditions is missing, so that Oracle has no choice but to perform a cartesian join. With large tables, this is a performance disaster.

It is possible to see a Cartesian Join in the explain plan where the query is logically correct, but I associate this with older versions of Oracle.

Also look for the unused compound index. If the first column of a compound index is not used in the query, Oracle may use the index inefficiently, or not at all. Let me give an example:

The query was:

```
select * from customers
where
     State = @State
     and ZipCode = @ZipCode
```

(The DBMS was not Oracle, so the syntax was different, and I've forgotten the original syntax).

A quick peek at the indexes revealed an index on Customers with the columns (Country, State, ZipCode) in that order. I changed the query to read

```
  select * from customers
   where Country = @Country
      and State = @State
      and ZipCode = @ZipCode
```

and now it ran in about 6 seconds instead of about 6 minutes, because the optimizer was able to use the index to good advantage. I asked the application programmers why they had omitted the country from the criteria, and this was their answer: they knew that all the addresses had country equal to 'USA' so they figured they could speed up the query by leaving that criterion out!

Unfortunately, optimizing database retrieval is not really the same as shaving microseconds off of computing time. It involves understanding the database design, especially indexes, and at least an overview of how the optimizer does its job.

You generally get better results from the optimizer when you learn to collaborate with it instead of trying to outsmart it.

Good luck coming up to speed at optimization!

Share Improve this answer Follow

edited Oct 25, 2008 at 13:25

answered Oct 25, 2008 at 13:17

Walter Mitty

18.9k ● 2 ● 31 ● 59

Thanks for the advice. I had also ran into the Cartesian Join issue you mentioned above in a few other queries a was working on and that made a huge difference in time and now the queries actually return what they are supposed to. Go figure. – Jacob Schoen Oct 26, 2008 at 20:28

8

You get more than that actually depending on what you are doing. Check out this explain plan page. I'm assuming a little bit here that you are using Oracle and know how to run the script to display the plan output. What may be more important to start with is looking at the left hand side for the use of a particular index or not and how that index is being utilized. You should see things like "(Full)", "(By Index Rowid)", etc if you are doing joins. The cost would be the next thing to look at with lower costs being better and you will notice that if you are doing a join that is not using an index you may get a very large cost. You may also want to read details about the explain plan columns.

Share Improve this answer Follow

edited Oct 24, 2008 at 18:15

answered Oct 24, 2008 at 18:04

carson
5,759 ● 3 ● 26 ● 25

I appreciate you help, and especially the links. It is starting to make since to me now. Thanks again for the help.
– Jacob Schoen Oct 24, 2008 at 18:19

1 Joins not using indexes may be bad, they may be the absolutely best. It all depends. do not, do not, do not try to eliminate every full table scan with indexes. – Mark Brady Oct 24, 2008 at 20:11

You got the fuzzy end of the lollipop.

6

There is absolutely no way, in isolation, without a ton of additional information and experience, to look at an explain plan and determine what (if anything) is causing less than optimum performance. If query tuning could be reduced to a 10 step process it would be done by an automated process. I was about to list all of the things you need to understand to be effective at this but that would be a very long list.

the only short answer I can think of... is look for steps in the plan that are going through way more bytes than you'd guess. Then think about how you can reduce that number... via an index or partitioning.

Seriously, get Jonathan's Lewis book on Cost Based Oracle Fundementals

Get Tom Kyte's book on Oracle database Architecture and rent a cabin in the woods for a few weeks.

Share Improve this answer Follow

edited Oct 24, 2008 at 20:10

answered Oct 24, 2008 at 20:01

Mark Brady

I had started to get that feeling that this was not as simple as it was originally described to me at work. Thanks for the book suggestions, they will be added to my list of books to read. – Jacob Schoen Oct 26, 2008 at 20:17

1 Don't just add them to the queue... move them to the top and do whatever you can to read them. I'd say Lewis' book first. It's entirely about what an explain plan means -- although it's not written that way. – Mark Brady Oct 27, 2008 at 14:54

4

This is a massive area of expertise (aka a black art).

The approach I generally take is:

1. Run the SQL statement in question,
2. Get the actual plan (look up dbms_xplan),
3. Compare the estimated number of rows (cardinality) vs actual number of rows. A big difference indicates a problem to be fixed (e.g. index, histogram)
4. Consider if you can create an index to speed part of the process (generally where you conceptually think the plan should go first). Try some indexes.

You need to understand the O() impacts of different indexes in the context of what you are asking the database. It helps you understand data structures like b-trees, hash tables etc. Then, create an index that might work and repeat the process.

If Oracle decides not to use your index, apply an INDEX() hint and look at the new plan. The cost will be greater than the plan it did choose - this is why it didn't pick your index. The hinted plan might lead to some insight about why your index is not good.

Share Improve this answer

Follow

answered Oct 25, 2008 at 8:20

WW.

24.3k ● 15 ● 97 ● 124

Oracle not choosing to use an index seemed strange to me at first, until your explanation above, and now I realize I am in deeper than I ever realized. If only they would get us an experienced DBA to work with us, we would be better off.
– Jacob Schoen Oct 26, 2008 at 20:24