Bind Variable Peeking: Bane or Boon?

Christian Antognini
Trivadis AG
Zurich, Switzerland

Since its introduction in Oracle9i, many people have argued over the pros and cons of bind variable peeking. Not surprisingly, however, most of the contributors to this debate have mostly pointed out the disadvantages of the feature. This is probably because it's much easier to recognize when something goes wrong because of a specific feature than the opposite. So, before sharing my answer with you on the question of "Bane or Boon?", I would like to briefly review what the pros and cons of bind variable peeking are. To do so, however, I have to begin with a discussion of the pros and cons of bind variables. This is necessary because too often I hear people criticize bind variable peeking when, in fact, it's the utilization of bind variables that should be questioned.

From a performance point of view, bind variables introduce both an advantage and a disadvantage. The advantage of bind variables is that they allow the sharing of cursors in the library cache and that way avoid hard parses and the associated overhead. The disadvantage of using bind variables in WHERE clauses, and only in WHERE clauses, is that crucial information is hidden from the query optimizer. For the query optimizer, it is in fact much better to have literals instead of bind variables. With literals, it is able to improve its estimations and, therefore, to choose optimal execution plans. This is especially true when it has to check whether a value is outside the range of available values (that is, lower than the minimum value or higher than the maximum value stored in the column), when a predicate in the WHERE clause is based on a range condition (e.g. HIREDATE > '2009-12-31'), and when it makes use of histograms. As a result, for cursors that can be shared, you should always use bind variables if one of these three conditions is not met. The main exception is for SQL statements for which the parsing time is several orders of magnitude less than the execution time. In this kind of situation, using bind variables is not only irrelevant for the whole execution time, but it also increases the risk that the query optimizer will generate very inefficient execution plans.

To overcome the disadvantage due to bind variables, Oracle introduced *bind variable peeking*. The concept of bind variable peeking is simple. During the physical optimization phase, the query optimizer peeks at the values of bind variables and uses them as it would with literals. Hence, at first sight, we have a win-win situation: the number of hard parses is reduced and the query optimizer makes better estimations. The problem with this approach, however, is that the generated execution plan depends on the values provided by the first execution. Consequently, as long as the cursor remains in the library cache and can be shared, it will be reused. This occurs regardless of the efficiency of the execution plan related to it. As a result, some executions might be efficient and some others might not. In addition, depending on how long a cursor remains in the library cache, unpredictable response times might be experienced. To solve (or at least diminish) the disadvantage due to bind variable peeking, as of Oracle Database 11g, a new feature called *extended cursor sharing* (also known as *adaptive cursor sharing*) is available. Its purpose is to automatically recognize when the reutilization of an already available cursor leads to inefficient executions. This is a good thing. But, be careful, the database engine is only able to recognize such a situation when a given cursor is executed several times with a suboptimal execution plan. In other words, the database engine tries to learn from its mistakes (but, of course, it has to make some mistakes to be able to recognize them).

In summary, to increase the likelihood that the query optimizer will generate efficient execution plans, you should not use bind variables when one of the cases mentioned before are met. Even if bind variable peeking might help, it is sometimes a matter of luck whether or not an efficient execution plan is generated. The only exception is when the new extended cursor sharing of Oracle Database 11g automatically recognizes the problem.

So, bane or boon? As usual, it depends. There are applications that work very well with bind variable peeking. The reason is simple: they have been implemented carefully. They use bind variables in a sensible way. On the other hand, there are applications that experience unpredictable execution times for the opposite reason. For them, it is usually necessary to disable the feature through the undocumented initialization parameter _optim_peek_user_binds. Of course, it is never a good thing to set an undocumented parameter. It is my opinion that Oracle should not only promote this parameter as a regular one, but also provide hints to control the utilization of bind variable peeking. In other words, they should give us the ability to choose whether it is beneficial to use bind variable peeking or not.

Since 1995, Christian Antognini has focused on understanding how the Oracle database engine works. His main interests include logical and physical database design, the integration of databases with Java applications, the query optimizer and basically everything else related to application performance management and optimization. Christian is the author of the book Troubleshooting Oracle Performance (Apress, 2008). He is currently working as a principal consultant and trainer at Trivadis in Zurich, Switzerland.