

INDEX COLUMNS: WHY THE ORDER DOESN'T MATTER

Index Usage

All DBAs are familiar with the guidelines regarding index usage. We all understand that we should build indexes on columns that are very discriminatory—that is, they help Oracle to isolate the result set as quickly as possible. In addition, we understand that the *leading columns* of an index should match the search criteria specified in your Sql.

Given the above guidelines, it only seems natural, then, to conclude that the most discriminating columns should be placed before the other columns in the index. It just seems obvious that starting with the most discriminating column must reduce the work, because Oracle can more efficiently discriminate if it begins with the “best” column first. This conclusion fits right in with our practice of “leading edge” columns in an index.



Surprisingly, our intuition is wrong!

While it's true that the leading columns of the index must match the Sql criteria, once that condition is met, it doesn't make a difference which column comes first. Our intuition is flat-out wrong!

How Can This Be?

Tom Kyte, in his excellent book, *Expert Oracle Database Architecture: 9i and 10g Programming Techniques and Solutions*, discusses this issue. He proves, via test cases, that Oracle doesn't perform more work when a non-selective column is the first field in the index. Unfortunately, Tom doesn't really explain *why*.

The essence of the issue is simply this: Once Oracle has qualified an index to be used, there is nothing magical about which column on the “left” side. Oracle's algorithm, when building (or traversing) a branch of a B*Tree, *combines* the relevant columns. It's the *composite* that is used to point to the next branch of the B*Tree, not just one column. So “left” or “right” has no particular benefit.

An Example

Consider the following indexes:

INDEX1: (ZIP_CODE, GENDER)
INDEX2: (GENDER, ZIP_CODE)

Assume we are looking for the combination of Zip_Code = 94568 and Gender = 'Male.' Before Oracle traverses the index, it combines the two conditions. The composite value, something like *94568_Male* has the same discriminatory value as *Male_94568*.