Oracle SQL & PL/SQL Optimization for Developers Documentation

Release 2.1.1

Ian Hellström

October 03, 2015

Contents

CHAPTER 1

Introduction

SQL is a peculiar language. It is one of only a handful of fourth-generation programming languages (4GL) in general use today, it seems deceptively simple, and more often than not you have many quite disparate options at your disposal to get the results you want, but only few of the alternatives perform well in production environments. The simplicity of the language veils decades of research and development, and although the syntax feels (almost) immediately familiar, perhaps even natural, it is a language that you have to wrap your head around. People coming from imperative languages often think in terms of consecutive instructions: relational databases operate on sets not single entities. How the SQL optimizer decides to execute the query may not coincide with what you think it will (or ought to) do.

To the untrained eye a database developer can seem like a sorcerer, and it is often said that query tuning through interpreting execution plans is more art than science. This could not be further from the truth: a thorough understanding of the inner workings of any database is essential to squeeze out every last millisecond of performance. The problem is: the information is scattered all over the place, so finding exactly what you need when you need it can be a daunting task.

1.1 Why This Guide?

While it's easy to write bad code in any programming language, SQL — and to some extent PL/SQL too — is particularly susceptible. The reason is simple: because of its natural-looking syntax and the fact that a lot of technical 'stuff' goes on behind the scenes, some of which is not always obvious to all but seasoned developers and DBAs, people often forget that they are still dealing with a programming language and that SQL is not a silver bullet. Just because it looks easy does not mean that it is. We don't want to step on anyone's toes but frequently production SQL code (e.g. reports) is created not by developers but business users who often lack the know-how to create quality code. It's not necessarily their jobs to come up with efficient queries but it is not uncommon to hear their gripes afterwards, once they discover that a report takes 'too long' because the database is 'too slow'. The fact of the matter is that they have run into one of many traps, such as non-SARGable predicates, bad (or no) use of indexes, unnecessarily complicated (nested) subqueries that not even their creator can understand after a short lunch break but somehow magically deliver the correct, or rather desired, results. Other programming languages, especially the more common third-generation ones, do not have that problem: applications are mostly developed by professional developers who (should) know what they're doing.

There are many excellent references on the topic of SQL and PL/SQL optimization, most notably Oracle's own extensive documentation, Tom Kyte's Ask Tom Q&A website, entries by Burleson Consulting, Tim Hall's Oracle Base pages, Steven Feuerstein's PL/SQL Obsession, Oracle Developer by Adrian Billington, books, and a wealth of blogs (e.g. by Oracle ACEs).

'So why this guide?' we hear you ask. Two reasons really:

1. It's a long and at times arduous journey to master Oracle databases, and it's one that never ends: Oracle continues to develop its flagship database product and it is doubtful that anyone knows everything about its internals. We hope that other developers will benefit from the experiences (and rare insights) chronicled here. These page are