



Exploring the Return on Investment from *MIN-MAX PL/SQL*

Steven Feuerstein
steven@stevenfeuerstein.com
PL/Solutions
Blastoffplsql.com
September 1st, 2003

This document addresses the following important question:

What is your return on investment for sending a PL/SQL developer or DBA to the *MIN-MAX PL/SQL* seminar?

Budgets are very tight, so you can't afford to send your employees to all the trainings they want to attend. You have to choose carefully, and your choice needs to be driven primarily by a desire for the shortest-possible impact of those trainings on your development efforts.

This paper explains how attendance at the *MIN-MAX PL/SQL* seminar can pay for itself and more in less than three months. Let's take a look.

First of all, the seminar fee is \$1250 for three days (you can actually pay as little as \$1000, but let's assume the high end for expenses). Of course, this means that your employee is also not working for three days. Let's assume a cost of \$100 per hour for your employee. For three eight-hour days, that costs you another \$2400 dollars in lost productivity. So the total cost of the seminar is probably something like \$3,650. Let's round up to \$4,000.

Note: **If you are attending a one-day seminar**, the cost of the seminar to you (including time away from work) will be approximately \$3000 at the rates used above.

How can MIN-MAX PL/SQL help you recover and exceed \$4,000 in expenses?

MIN-MAX PL/SQL spends very little time introducing students to new features of the PL/SQL language. Instead, we focus on programming techniques that follow widely-recognized best practices. Why should you care about best practices? Because by following carefully selected and prioritized best practices, you can:

- ? Minimize the number of bugs that developers introduce into the code, and therefore the amount of time and effort needed to test and fix the code.
- ? Minimize resources needed to maintain your applications once they go into production, because well-written software is much easier to read, understand and change.

Well, that's a lot of general stuff. Let's get down to specifics and how your budget can clearly benefit. Let's take a look, in fact, at one of the most critical aspects of PL/SQL development: writing SQL.

MIN-MAX PL/SQL shows developers how to tightly control the dispersal and inevitable redundancy of SQL statements in PL/SQL programs. As you are probably aware, SQL coding, debugging and tuning can take up a very large percentage of total development time. Most development organizations do not establish standards or controls over how individual developers write the SQL in their applications. As a result, many developers write the same or similar statements over and over again, with inconsistent error handling. It is very difficult to build, tune and maintain such code.

Upon return from **MIN-MAX PL/SQL**, developers will be able to easily shift from "hard coding" queries and DML statements to *generating* those statements into reusable packages. **MIN-MAX PL/SQL** provides the generator utilities that do all the work (they are called Encapsulation Generators because they encapsulate or hide the SQL behind program interfaces).

These utilities generate code designed by Steven Feuerstein, who is one of the world's leading authorities on the PL/SQL language (and teacher of the seminar); needless to say, the code conforms to best practices and is optimized within the PL/SQL language for best performance. Note: the Encapsulation Generators come free with the course, which also shows developers why they are important and how to use them.

Suppose your developer uses the Encapsulation Generators for a single critical table, against which she or he needs to write four different queries and a half dozen inserts, updates and deletes.

Here are optimistic estimates to write, test, debug, and maintain "hard-coded" SQL for those 10 statements.

Task	Time in Hours
Write: quick and dirty, putting the SQL directly into each program as needed. Average of fifteen minutes per statement.	2.5 hours
Test: construct crude test scripts and run them with a minimum of data variations. Average of half hour per statement.	5 hours
Debug: assume minimal number of errors (BIG assumption!)	1 hour
Maintain this body of code: as soon as any change occurs to the table, it will be necessary to hunt down all the SQL statements and change them, then re-test and debug. Let's say one hour of hunting, and then a repeat of the above.	9.5 hours
Total	17.5 hours

Of course, you have more than one table in your application. You probably have, in fact, more than twenty tables. Suppose, however, that you only need to write intensive SQL with five of these tables. We then have an estimate of **87.5 hours** to perform the above operations "manually."

Hopefully you will agree that this is a very realistic, perhaps optimistic projection.

Now, if the developer uses one of the Encapsulation Generators, I offer these estimates:

Task	Time in Hours
1 Learn: get familiar with how the generator works	2 hours
2 Write: Rather than write anything, you will run the generator. You might need to do this several times to get it "right the first time."	1 hours
3 Test and debug: One enormous advantage of generated code is that once the template is stabilized and debugged, you are working with pre-tested, bug-free code.	1.5 hours
4 Debug: the probability of the presence of a bug goes down dramatically with generated code.	.5 hours
Maintain this body of code: as soon as any change occurs to the table, the developer will simply re-generate the encapsulation code and repeat the above steps. Since the details of the SQL have been hidden behind the programmatic interfaces, little or no change is required to the existing application code. So: one hour for changes to invocations of encapsulated code, then repeat steps 2-4.	4 hours
Total	8.0 hours

Applying this estimate to the five tables mentioned previously (and *not* repeating the two hours of learning, since that is only done once for the utility and not for each table), we then come up with an estimate of **40 hours** to perform the above operations with generated and tightly controlled SQL code.

Savings to your organization in time: 47.5 hours. At \$100 per hour, this comes to \$4,750.

These are, of course, the savings accrued in just the first round of development and maintenance; the cycle will repeat itself many times over a single year, resulting in much higher savings.

Conclusion: simply changing the way developers write and maintain SQL statements inside PL/SQL applications quickly justifies the participation of your developers and DBAs in **MIN-MAX PL/SQL**.