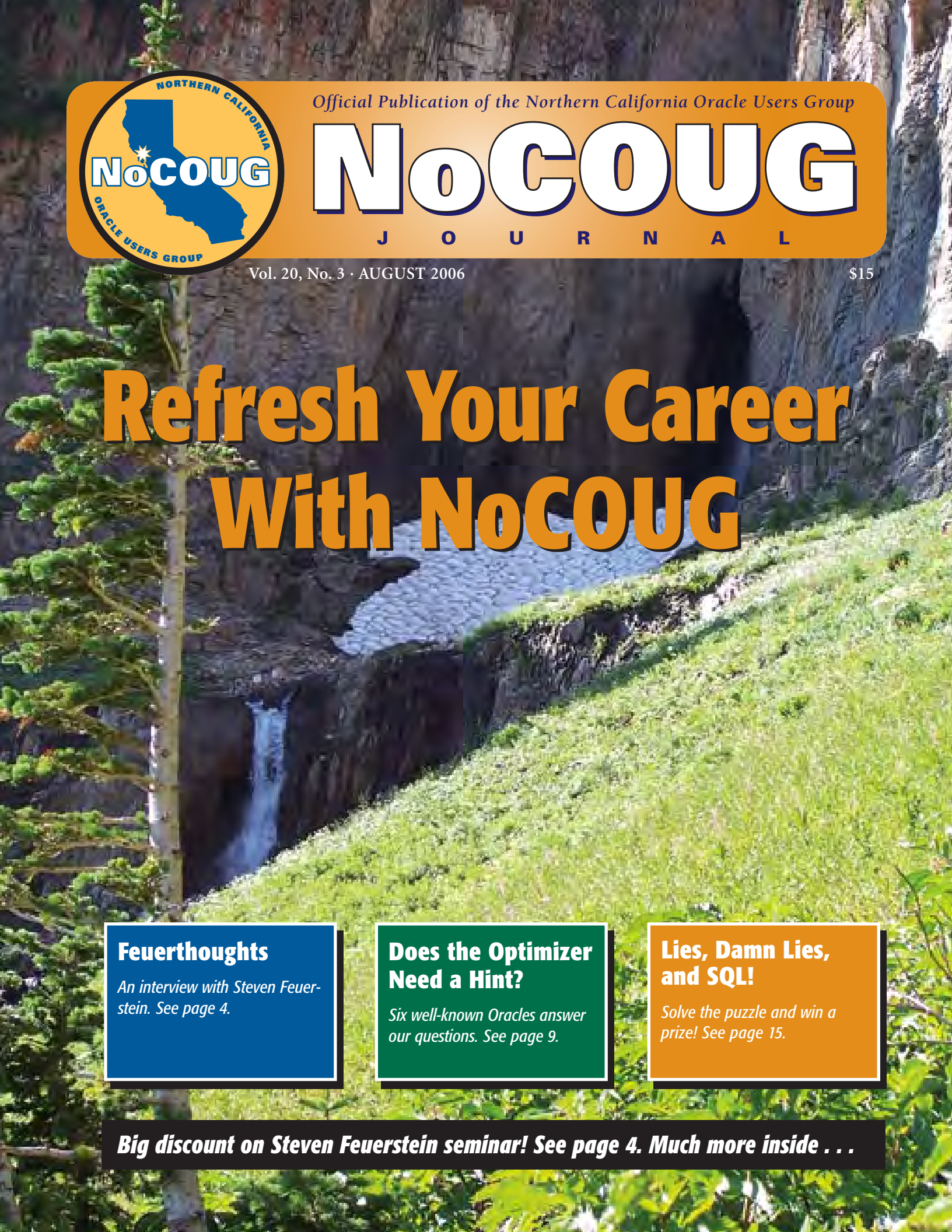
Official Publication of the Northern California Oracle Users Group

NoCOUG JOURNAL

Vol. 20, No. 3 · AUGUST 2006 $15



Refresh Your Career With NoCOUG

Feuerthoughts

An interview with Steven Feuerstein. *See page 4.*

Does the Optimizer Need a Hint?

Six well-known Oracles answer our questions. *See page 9.*

Lies, Damn Lies, and SQL!

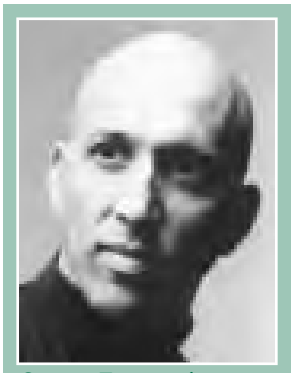
Solve the puzzle and win a prize! *See page 15.*

Big discount on Steven Feuerstein seminar! See page 4. Much more inside . . .

Feuerthoughts

An Interview with Steven Feuerstein

He couldn't move a mountain—nor pull down a big old tree
But my daddy became a mighty big man with a simple philosophy.
Do what you do, do well boy—do what you do, do well;
Give your love and all of your heart—and do what you do, do well.



Steven Feuerstein

NoCOUG welcomes Steven Feuerstein, considered one of the world's leading experts on the Oracle PL/SQL language, having authored or co-authored ten books on the subject—as well as a new book on MySQL! Steven currently serves as PL/SQL evangelist for Quest Software. He authors the popular PL/SQL Best Practice *column on the Oracle Technology Network. Steven is now busy building a unit testing tool for PL/SQL developers, code-named Qute—the Quest Unit Test Engine.*[1]

On Being an Oracle Guru

What does an Oracle guru do? Do you teach? Travel? Consult? Hobnob with captains of industry? Or do you hold down a day job like the rest of us?

Hmmm. Well, first I've got to tell you that I am not terribly comfortable with the "guru" thing. Mostly, I am just a very lucky guy who wrote the right book at the right time. After *Oracle PL/SQL Programming* was published in September 1995 and started selling like hotcakes, I was able to end my career as an hourly consultant. That meant in turn that I could devote more time to studying and playing around with PL/SQL. It's amazing what you can accomplish when you don't have a boss telling you what to do all the time!

[1] **www.unit-test.com**

So what do I do? These days, I spend virtually all my available time developing software to help programmers unit test their code. I am very excited about the impact that Qute—which we expect to release in October 2006—will have on the PL/SQL world. Beyond that, I do travel a fair amount, mostly doing trainings and seminars. My favorite venue is an Oracle User Group program. Oracle Corporation in Europe also likes to invite me over to train their customers. I spent two weeks in May skipping around through Scandinavia and the Baltics—Copenhagen, Stockholm, Oslo, Riga, and Tallinn in just one week—and finishing up in Prague, which is quite a beautiful city.

You certainly didn't become an Oracle guru overnight. Or did you? Where did you begin? How did you get to this stage?

I have two confessions to make: first, I have got to be one of the most narrowly specialized technologists in the world. I really just know one thing: PL/SQL. Fortunately, I know that pretty well—and fortunately also, Oracle has done a splendid job of designing, building, and enhancing the language. Sure, I used to write Fortran, but these days if you ask me about anything outside of PL/SQL you will draw a blank. Second confession: I am not really much of a technologist. I took three "101" courses on programming in college, and that is it for formal education. I have come to recognize over time that my strength is not that I am an amazing computer scientist sort of guy, but that I am an effective communicator. People seem to be able to actually read my stuff, understand it and then learn from it. And it hasn't hurt, from what I am told, to let my sense of humor appear in my books . . . .

OK, having said that, I got my beginning with Oracle as a presales person. I spent a couple of years following salespeople around to accounts and doing the dog-and-pony shows for them with SQL*Plus, SQL*Forms, etc. I can still remember my first public seminar for Oracle. Basically, what we were doing back in 1987 was running SQL*Plus scripts and saying, "Isn't this language amazing?" So I was given the standard script from Oracle for the presentation, drove up to Milwaukee with my boss, John Cordell, and got up in front of about 30 people to wow them. Now, I am a compul-

sive tinkerer, so I decided that there were some ways I could improve the script. Unfortunately, I am also not the most disciplined person so there were two bugs in this canned script! It was quite embarrassing, but I survived, and John was very nice about it.

Presales was interesting and sometimes exciting, but programming was way better, so I would constantly dabble with the Oracle tools, building little apps for my co-workers to use. This developed into TeamSell, a sales support application that caught the eye of Mike Fields, head of U.S. Sales in the early '90s. I was drafted to join a small dev team and we started building some very cool SQL*Forms apps to support the U.S. sales force. Then Larry canned Mike, and I was told to go back out on the road to help sell Oracle. I said no thanks, took the first consulting job I was offered, and ended up spending three years at McDonald's headquarters in Oak Brook. While there, I saw an appeal on CompuServe for Oracle authors and I thought "Why not? I can write." So I wrote that first book, which was the first independent text on PL/SQL, and it changed my life. From that point on, I was virtually a full-time student of the PL/SQL language, researching, writing, teaching, building code, etc. As I mentioned earlier, once you can structure your own time, all it takes is discipline—and reasonably good typing skills—and you can accomplish an awful lot!

Some people can perform seeming miracles with straight SQL, but the statements end up looking like pretzels created by somebody who is experimenting with hallucinogens.

I'm sure you have many interesting stories to tell about your travels and interactions. Tell us a story!

I am very pleased to have been so successful with PL/SQL. I truly have a wonderful life as a result. And I am sure that many companies have benefited from my writings. But what I find most rewarding is the impact I have had on the lives of individual programmers. It is not terribly uncommon to have a person walk up to me at a conference or training and tell me something like the following—which did happen in Chicago in 1998: "I was a union electrician at a steel mill in northern Indiana. When I got laid off, I went back to school and studied Oracle programming. I got hold of your book and it changed my life. Now I have a great job, my wife doesn't have to work and can raise our kids." Ah, that is so satisfying!

We like stories! Tell us another story!

I have traveled to something like 25 different countries, sharing my thoughts on PL/SQL. Whenever I go to a new city, I like to seek out the finest art museum and soak up the creative energies of the artists. I feel privileged to have been able to see so many incredible works of art over the years. I think that the impulse to create is one of the things that make humans special. I was in Prague in late May 2006 and took an extra day past my trainings to wander the city. Prague is a beautiful place and I encourage you all to visit, but what I found most incredible were the *sidewalks*. I am used to simple concrete slabs. In Prague, you will find that many sidewalks are actually mosaics made of two inch cubes of quartz and other rock, arranged in the most elaborate and intriguing patterns. You can read more about—and see—these sidewalks by visiting my blog.[2]

On PL/SQL

Why PL/SQL? Why not JDBC or SQLJ? What are the intrinsic advantages of PL/SQL? Flexibility? Efficiency? Can you give us some examples? Don't compiled languages have the advantage of speed over interpreted languages like PL/SQL?

Why PL/SQL? Why is my name Steven? Why do I live in Chicago? There are solid explanations for each of these, but it mostly comes down to: this is where my life has led me. I have made very few conscious career decisions in my life. I didn't seek out Oracle. I was just looking for another programming job, and it turned out the headhunter was filling presales job slots—my biggest concern about accepting the job was that I would have to wear a suit and tie each day. I sure didn't plan on becoming a "guru"—I just wrote stuff that people liked.

Having said that, PL/SQL is a very wonderful language. It is so simple and easy to use, and yet very powerful. It is certainly not as powerful or complete as Java, and there aren't nearly as many developers writing PL/SQL as Java, but it gets the job done. I like to call PL/SQL the "Cobol of the 21st century." It may not be terribly glamorous and it probably won't be used very much for new development 20 years from now, but we have used it to implement mission-critical production apps that run tens (hundreds?) of thousands of businesses around the world.

And 20, 30 years from now, programmers will still need to maintain and enhance those apps. Which is why it is *so important* to build comprehensive regression tests for the code we write today.

SQL is a set-oriented non-procedural language; i.e., it works on sets and does not specify access paths. PL/SQL on the other hand is a record-oriented procedural language, as is very clear from the name. What is the place of a record-oriented procedural language in the relational world?

Its place is proven: SQL is not a complete language. Some people can perform seeming miracles with straight SQL, but the statements can end up looking like pretzels created by someone who is experimenting with hallucinogens.

We need more than SQL to build our applications, whether it is the implementation of business rules or application logic.

PL/SQL remains the fastest and easiest way to access and

[2] **feuerthoughts.blogspot.com**

manipulate data in an Oracle RDBMS, and I am certain it is going to stay that way for decades.

Some might worry that using PL/SQL would chain them to Oracle and restrict their future options. What would you say to those you who wish to create database-agnostic applications?

Well, like I said earlier, PL/SQL is all I know, so I automatically self-select into working with and talking to people who *have* committed to both Oracle and maximizing Oracle technology.

If you want to create truly DBMS-independent code, you should at least take the time to write a complete layer of code, whether you call it a table API or a set of services, that *hides* the database activity. Then within that data access layer, you can write code that connects to one or more databases.

What development shops should work very hard to avoid, though, is writing any SQL statements in front-end logic. My feeling is that Java (and .Net and VB and . . .) programmers should *never* write SQL. They generally don't have much respect for the language and don't do a very good job writing it. Also, if you put your SQL in the application layer, it is much harder to maintain and optimize.

You write books, but you also create products and resources for PL/SQL developers. Tell us about some of your innovations, such as 21st Century PL/SQL and PL/Generator.

I like to create things. Sometimes this creative process is an excellent use of my time and sometimes . . . well, you know that joke about a tree falling in the forest? If you write a program and no one ever uses it, did you really write that program?

I have probably written more code that no one will ever run than most people have ever written, period.

So, yes, I have been creating all sorts of stuff over the years, ranging from products to resources. My first serious product effort was XRay Vision, a debugger for SQL*Forms V3. It was very cool stuff, written in SQL*Forms itself. Unfortunately, it came out when the product was on its last legs.

Since then, I have worked on all sorts of stuff: the original RevealNet Knowledge Base on PL/SQL (an interactive and very active form of information, now offered as the Knowledge Xpert™ from Quest); CGML, the Code Generation Markup Language, handy for describing design patterns; PL/Generator, a tool that generated comprehensive table APIs; PL/Vision, a library of some 1,000 PLSQL procedures and functions (now available as freeware on the Quest Pipelines[3]); and 21st Century PL/SQL, a webpage that collects ideas from the development community on how to improve the PL/SQL language.[4]

In the last few years, I have focused my creative juices on two tools: (1) Qnxo (Quality iN, eXcellence Out)[5], a design pattern factory tool that not only generates table APIs, but offers a set of about 700 scripts for PL/SQL programmers—I *hate* to reinvent the wheel; and (2) Qute (Quest Unit Test Engine), a unit-testing tool which allows you to describe through a UI the tests you want to run, and then generates the test code, runs the tests, and shows you whether or not your code worked, and if not, where it failed.

Quest Software just acquired both Qnxo and Qute. You will see Qute this fall at Open World with a new name and incredible functionality. It is going to completely change the way people think about and perform PL/SQL testing. And then we will apply the same techniques to MySQL-, SQL Server- and DB2-stored programs.

Do you have a pet peeve about PL/SQL? Have the limitations of the DBMS_OUTPUT function been fixed yet?

I just *love* to complain about PL/SQL! All in good fun and all with the intention of improving the language, of course.

Yes, DBMS_OUTPUT was a very easy target for my scorn. A maximum of 255 characters can be displayed? Ridiculous! And now, in Oracle Database 10*g* Release 2, it has been fixed: DBMS_OUTPUT.PUT_LINE will accept up to 32K of stuff to display.

Why did it take Oracle so long to do this—Chris Racicot, the PL/SQL Development Manager, told me that once they looked at the situation, they realized that all they had to do was change a single number! Argh! Because the PL/SQL team, while substantial in size, has a very long list of enhancements to work on, and many of them are still really big, complex, and absolutely critical—such as building an optimizer for the language, first released in Oracle Database 10*g* Release 1.

So things like DBMS_OUTPUT tend to fall by the wayside.

In another section of the* NoCOUG Journal*, we asked the question "Does the [Query] Optimizer Need a Hint" to a panel of performance experts. This is not directly related to PL/SQL, of course, but do you have an opinion on the matter that you would care to share?

Hmm. Well, I can always offer an opinion; it just might not be very well-informed. But in the United States these days, that almost seems to be a requirement to state an opinion, write laws, decide policy...oh, yes, PL/SQL, right.

Well, I am in many ways a very typical user of PL/SQL and Oracle technology generally: I just want it to work and work fabulously with an absolute minimum amount of attention

If you put your SQL in the application layer, it is much harder to maintain and optimize.

[3] **www.quest-pipelines.com**

[4] **www.oracleplsqlprogramming.com/IC/index.php**

[5] **www.qnxo.com**

and fuss on my part. So I am sure that the optimizer needs improvement and I look forward to that panel of experts helping Oracle figure out what is needed. For me, when I have a problem with my SQL, I use the tuning tools in Toad to identify ways to improve performance.

If I could only afford to buy one of your books, which one would you pick for me—and why?

Without a doubt—*Oracle PL/SQL Programming, Fourth Edition*. It is big, it is comprehensive and it is oh so much fun to read (or so people tell me, though I shudder when a programmer tells me that "I spent the whole weekend reading your book!").

If you can spare an extra $15 or so, I also suggest you pick up *Oracle PL/SQL Best Practices*. It is a concise presentation of my ideas for writing high-quality code. You will not, however, be able to incorporate *OPBP* into your fitness regime. It is too small and light. If you need to work on upper body strengthening, stick with *OPP4*!

What will we learn at your seminar?

After attending the 21st Century PL/SQL course, you will be aware of all the most important and interesting new features in the PL/SQL language. And much of that stuff is centered on PL/SQL collections, array-like structures that I consider to be the latest frontier and challenge for PL/SQL developers. If you don't know about or are not completely comfortable with collections, attend this seminar.

With collections, you can employ FORALL and BULK COLLECT to turbocharge your SQL statements, and you can build table functions that allow you to call a PL/SQL function as if it were a relational table. Great stuff!

Of course, you won't be an instant expert in applying the many features I will cover on that day. You will, however, have lots of new ideas, plus you will learn that you can always visit my PL/SQL portal[6], and download all of my training materials, along with supporting code, to follow up on the course with your own personal studies—feel free to copy and use all that code!

On the State of the Industry

All the jobs are going to India, Russia, and China. I know veteran IT professionals who now own health food stores, sell teddy bears on the Web, sell insurance, and patrol the streets of Berkeley on bicycles (bicycle cops). What's your advice for IT professionals? Should we give up and find other professions?

How do I feel about friends, coworkers, and others I meet in the U.S .struggling, when previously everything was golden? I feel awful about it.

I also feel great about people in India, Russia and China finally able to participate in the great world of software development, and thereby improve their own quality of life and those of their children.

I feel awful about friends, coworkers, and others I meet in the U.S. struggling, when previously every-thing was golden. I also feel great about people in India, Russia, and China finally able to participate in the great world of software development, and thereby improve their own qualify of life.

Software programming is a great profession and I think that anyone with training in the field should work hard to stay in it. Where else can you get a company to pay you to sit around and think about stuff, and write it down?

Yes, but how to hold on to those jobs? Speaking from my own experience, I would say that the trick is to specialize, to find a niche that fills a need that either not too many others are filling or know about, or is particularly challenging to do. If you can make yourself indispensable, you will find valuable and rewarding work.

Is Oracle facing a credible threat from open source databases such as MySQL and Postgres? Some even think that Oracle is trying to kill off the competition by purchasing InnoBase and Sleepycat. Computer Associates recently released commercial Ingres into the public domain—so a great deal of advanced technology is now available to open source developers. MySQL is working toward SAP certification and even wants to grab a piece of the data warehouse pie. What's the future of open source database technology? Why has it been so successful thus far? Will it continue its winning ways?

I do believe that database technology is becoming commoditized, at least on the low end—small databases, non-mission-critical applications, etc. This process will continue. Clearly, with enough time, even when people work without compensation on software, they can create amazing products.

It does seem like Oracle is trying to forestall and minimize the impact of MySQL, but I can't imagine that will work for very long—hey, that's part of the reason I co-authored a book on MySQL with Guy Harrison.

Oracle saw long ago that it needed to sell more than database technology, hence the move—with very mixed success to date—into applications.

As an owner of Oracle stock, I sure hope that Larry gets it right.

[6] **www.oracleplsqlprogramming.com**

The trick is to specialize, to find a niche that fills a need that not too many others are filling or know about, or is particularly challenging to do. If you make yourself indispensable, you will [always] find valuable and rewarding work.

Database technology has enabled the Information Age with all its wonders and benefits. On the negative side, our privacy and freedoms have been eroded as every detail of our lives from our shopping habits to our travel habits to our telephone calls is being captured in a database. The cell phone companies even have the ability to track us every minute of the day and may be storing every move we make in databases. How should society tackle this problem? Is legislation the answer?

Oh, my. That is a big question and requires an answer that far exceeds the scope of this interview. Fortunately, it is *precisely* the topic of my keynote presentation at NoCOUG's summer conference on August 17: *Software Programmers: Heroines and Heroes of the Twenty-First Century*.

Here's the bottom line—with ideas drawn liberally from Lawrence Lessig: Code, software, is a new form of law—human-created constraints on the behavior of other human beings.

We, software developers, have a responsibility to make sure that our software is used for beneficial purposes.

We write software; therefore we are in a sense lawmakers or enforcers of law. We, software developers, have a responsibility—individually and as a group—to make sure that our software is used for beneficial purposes.

Software developers should actively engage in whistleblowing, in challenging management on what they are building, when a corporation or government is using software in harmful ways.

More on that on August 17!

On Life Outside Work

What are your interests outside work?

Family, first and foremost, of course. I have been with my amazing, brilliant, and beautiful wife, Veva Silva, for 25 years. Our two kids, Chris and Eli, are wonderful human beings and both are artists—Chris is a well established muralist[7] and Eli is a jazz guitarist in the making.

Let's see. I turn 48 this September, which means my body is no longer the young, agile machine it used to be. So I spend a fair amount of time maintaining that machine—if you are having problems with your back, shoulders, neck, wrists, etc., get into a regular fitness routine. Most important: lots of abdominal exercises and stretching.

Beyond that, I have long complemented my software work with political activism. I spent years protesting our policies in Central America. Lately, I have gotten involved in Middle East peace issues. I serve as the president of the board of directors of the Refuser Solidarity Network[8], which educates the public about the Israeli "refusenik" movement.

Twenty or thirty years from now, I want to have some very good answers for my grandchildren when they ask me what I did during these years. So I get involved and do what I can. I urge all of you to do the same, according to your interests and beliefs.

I often wonder what advice I would repeat to my child if I had only a few minutes left on earth. We only have a few minutes left in this interview. What advice do you have for us—your fellow human beings?

As Spidey's uncle said, with great power comes great responsibility. We in the U.S., each one of us, actually do have great power. But we have largely abdicated responsibility. We should do everything we can to re-establish a real democracy, one that benefits the majority.

Our focus and priority should be on our children, the future of our species.

Finally and more generally, our focus and priority should be on our children, the future of our species. Make sure you spend lots of time with children, your own and others, enriching their lives, sharing your experience, giving them unconditional love and support. The world will be a far better place for it.

Thanks for providing me with this soapbox. I do so love to go on. ▲

Steven can be reached at **steven.feuerstein@quest.com**.

[7] **www.chrissilva.com**

[8] **www.refusersolidarity.net**