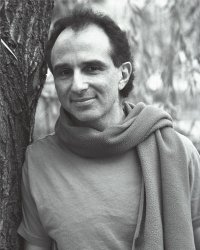
**Dennis Shasha Speaks Out  
on How Puzzles Helped Throughout his Career, Large Combinatorial Problems,   
Autonomic Computing, Dr. Ecco, Database Tuning, and More**

**by Marianne Winslett**

**

Dennis Shasha  
<http://cs.nyu.edu/shasha/>

*Welcome to ACM Sigmod Record’s Series of Interviews with Distinguished Members of the Database Community. I’m Marianne Winslett, and today we are in Providence, site of the 2009 ACM Sigmod/PODS Conference. I have with me Dennis Shasha, who is a Professor of Computer Science at the Courant Institute at New York University. His research interests include biological computing, database tuning, cryptographic file systems, and pattern recognition. He is a prolific writer, both within and outside of computer science. So, Dennis, welcome!*

*Let’s start off with what our readers really want to know: do you really wear a scarf and shorts every day, through both summer and winter?*

Yes, I’m afraid I do. In the winter time I get a lot of funny comments, like people ask me, “So, what are you drinking, man?” And one time I gave some money to a guy on the street, and he said, “You know, I should give you money, you should buy some long pants!” So it has been worth it just for the comments!

*Have you ever gotten frostbite?*

No, actually I am warmer wearing shorts, surprisingly, because the only part of me that gets cold is my hands. Because I wear shorts, I put my hands in my pocket. Because shorts are so thin, I get warmed by my legs, so I am actually warmer in shorts, than in long pants.

*What about the reverse, what about in summer when you’ve got your scarf on, how does that work?*

So, as I said, there are only a few parts of me that get cold. And one part is the neck, the other part is the hands. So it actually makes sense. Everything about me is a little bit strange, but it makes sense in that context.

*Have you ever gone any place really cold during winter?*

Well, Waterloo, Canada in February, and the waiter came down in the hotel and said, “Sir, you do realize we are in Canada, don’t you?” But it was fine.

*Biological computing: what’s in it for database researchers?*

I think it is a great field because computing, after all, started as an aid to science, and we are going back to that tradition when we help biologists, because biologists need a lot of help, actually, in all kinds of areas. They need help, above all, what they care about is experimenter time. They don’t care about computer time, per se, as long as it is not going to take them days. If it takes 20 minutes or 30 minutes, they don’t care. What they care about is, can they reduce the number of experiments they do, and still get more or less the same results. Can they use the experiments that they do to get better results? So the kinds of things that I do with biologists include helping them design experiments, so use statistics, combinatorial design, so that they don’t have to go through an entire search space but they can search a subset of that search space. Because, very many times, they have many variables they can modify. I work with plant biologists, they might add more nitrogen or less nitrogen, more carbon or less carbon, all these possibilities, the search space can be millions big, if you take all the possibilities. But using combinatorial design you can reduce it to a few hundred sometimes, and still get a very well sampled experimental space. And then on the analysis side, what is really nice, is that there is everything. There is some machine learning, data mining, there is statistics, and there is a lot of simulation. So there is a lot that computer scientists can do, and if a database researcher is willing to be all of computer science to the biologists then there is a lot to be done.

*Should we be training the biologists differently, do they need to know more computer science than they are learning, so they can do it themselves?*

I think biologists do quite a bit themselves, they do a lot of the tools themselves. So for example, there is this tool blast for comparing sequences. All biologists know how to use blast. But on the other hand, there are also just things that the computer scientists need to be doing. Those things often have to do with either a new kind of analysis that the biologists wouldn’t be able to do by themselves, or just looking at the data differently, or sometimes even helping the biologist design a better experimental program. So one of the biologists that I work with is a women named Gloria, and Gloria has been working on a certain kind of network on nitrogen uptake in plants. Well, what she does is great, but what we would really like to do is we would really like to take this network and be like electrical engineers and be able to reverse engineer it. That is what we would really like to do.

*What do you mean when you say reverse engineer it?*

Well, we’d like to find out which genes are connected to which other genes. As if we were probing an electrical circuit and finding out which circuit elements are connected to which other circuit elements. In doing that, what you do when you are an electrical engineer, you put current into a circuit element, or maybe you detach a wire; well, we can’t exactly detach a wire, but what we can do is we can make genes overexpress, so we can make them produce more protein than they would normally, and then we can see the effects of that. And so by doing that, we can actually march through the circuit, in principle, and find out exactly what is happening. So I find that very exciting, because Gloria and her lab have bought into this idea, and they are starting to do those kinds of experiments, and so it is a wonderful collaboration.

*But don’t you get all sorts of side effects once that one gene starts producing too much protein?*

Sure, but that is exactly what you want to find out, because you find out the side effects, and you can find out which other genes that gene is affecting. So that helps you figure out, “Ah-ha, that gene must be connected to the other gene!”

*It sounds like a close partnership that you are talking about.*

Yes, it is really a great partnership, we have a meeting every week, and we have cookies, and we make a lot of mistakes in the other field… One time I called alanine Alabama, because it had the “ala” prefix and I just forgot. So we really have a good time together.

*Dennis and Cathy Lazere wrote a book of biographies of great computer scientists, called* [*Out of Their Minds: the lives and discoveries of 15 great computer scientists*](http://cs.nyu.edu/shasha/outofmind.html)*. Tell us a story from that book.*

When we wrote Out of Their Minds, we thought, this is a great time to write about the seminal thinkers of computer science. We are not talking about Alan Turing, of course, but we are talking about the people who have made computing practical in a technological sense. So, we chose people who had already won the Turing Award, we chose others who soon won the Turing Award after we interviewed them, although I am sure there was no causal connection, so we started with algorithmists, like Michael Rabin, Dijkstra, Knuth, Lamport, and Cook and Levin for complexity theory. We interviewed languages people, McCarthy, Alan Kay, and Backus. And then we interviewed computer architects and we interviewed people in artificial intelligence. I avoided the database field because I felt that if I (probably it was a mistake) had interviewed one person, maybe other people would have been upset, so I avoided the database field, but that would have been good too. But anyway, a typical story from that book was Backus. Backus in some ways personifies a very unusual kind of researcher who nevertheless has the emotive influences that drive many researchers that I know. Mainly, he was easily angered. So, he had this very mixed, not very studious, childhood. At 25, he was graduating from the general studies program at Columbia, with a math degree, not knowing what he wanted to do. He got a job at IBM, almost by accident, because he was taking a tour, and somebody said, “You can interview here if you want. You can maybe get a job.” He took a test and they hired him. Then he was at programming in machine language, which he found horrible, so he started, what became the predecessor of FORTRAN, which was a way of doing floating-point arithmetic. When he proposed FORTRAN, he said “I am doing it because I find programming is too tedious, it is too difficult for people.” Interestingly enough, he was opposed in this desire by John Von Neumann, who thought that programming really wasn’t that hard. But Von Neumann was not your average person, and Backus understood this, and the IBM management understood that, and so, when he started the FORTRAN project, he was able to get the support of the IBM management who understood that if they could make software programming easier, then people would buy more computers. And that is what happened. Later, Backus went and participated in the ALGOL discussions, and he said it was very frustrating, people would just give an example for this, give an example for that, and nothing was getting anywhere. And so he developed Backus-Naur form, which is really very similar to the Chomsky notation for context-free grammars, just to make things clearer. So just because he was annoyed, he did a lot of his inventing.

*You make it sound like he is the accidental computer scientist.*

I think he is not accidental, because I think a lot of invention is annoyance. It is true, he is accidental in the sense that he might have gone into another field, because he wasn’t particularly, I mean, he was a math major, but he wasn’t an outstanding math student, by any means. So he is accidental in that sense, but I have a colleague, David Mazières, who is at Stanford now, and he is also very motivated by anger. It is a fun anger, it is kind of enjoyable to watch. He gets angry, and then he does something about it. He builds a better system.

*So what angers you?*

Well, I am not one of those kinds of people. What I like are hard problems, that I feel will really change things. Of course, it doesn’t always work, but sometimes it does.

*So have you had a favorite hard problem in the past?*

I have had a couple that I really enjoyed. One was this work that we did in tree matching. The reason that was fun was that this is work that started because a student of mine came to me, Kaizhong Zhang is his name, and he said, you know, this journal paper claims to have a really fast algorithm for this problem, but the algorithm doesn’t actually work. And so we looked at it together, and indeed, it didn’t work. And just seeing that, and seeing how sometimes a problem that seems closed because somebody else has solved it is not really closed because in fact it is not solved, can be a fun way to start a research project. And that project on tree matching, led to other projects on graph matching, and led eventually to my work with biologists because one of the applications was called RNA secondary structure. It was sort of this bunch of accidental coincidences of meeting people, getting these problems understood and used, and getting the algorithms used that lead me from one research area to the other.

*Dennis is also a puzzle column writer, for Dr. Dobbs’ Journal and for Scientific American. What’s that like?*

Well, Dr. Dobbs stopped a couple of years ago, and Scientific American is going through economic problems, so it is not clear that is going to continue, but what is has been like, is it has been fantastically fun. Because, first of all, you write puzzles and people send in solutions. And when people send in solutions, you become friends with them. And so I have what I call my “puzzle brain trust”, because sometimes I invent a puzzle that I can’t solve very well, so I send it out to various members of the puzzle brain trust, and sometimes they come up with much better solutions than mine. And it has been a remarkable experience, because I haven’t met my puzzle brain trust, in fact, I just discovered a couple of years ago that one of them happens to be deaf, and I had no idea. It has been just a wonderful experience in terms of the intellectual back and forth with many of the people I have communicated with. And nothing delights me more than finding somebody who has a better solution than I do, and it’s been also interesting, because it has helped me formulate problems for the biologists that I work with because sometimes the discipline of making a puzzle out of a problem can help you simplify the problem to its essentials and when you do that, you really start to understand it, and then you start to ask the right questions, and then sometimes you find a better solution. So the combinatorial design that I use for biologists, it turned into a safe cracking problem as a puzzle, where instead of a dial, you have many switches that are three way switches, and the question is: can you open the safe where if two switches (and you don’t know which ones) are in a certain configuration the safe will open. It’s a combinatorial design problem, and you explain it that way to biologists, and they understand it, mainly because they secretly want to break into safes, it’s possible!

*So when you publish these puzzles, do you always already have a solution?*

Oh, yeah, I already have a solution, but sometimes it is not the best solution. Because many of these problems are either continuous or some of them are NP complete, and so it is difficult. I did one puzzle about Zambonis. Zambonis are these machines that, after people have ice skated for a while, they clear off the ice. About a week ago, I received an email from the Zamboni people, they said, “We would definitely like to find the solutions to this puzzle.” But what the puzzle was, was that you have an ice rink, which is abstracted to a bunch of points more or less in an ovular shape, and that the Zamboni machine has to pass over every one of these points, but cannot turn at more than a 45 degree angle. And the question is, how do you make it go through all those points in as short a time as possible, so that people can get skating again? Because I was observing the Zambonis and although the drivers seemed to be having fun, they definitely were not doing the most efficient route. So that was a puzzle where you could ask that question, I suppose with enough computing power you could solve it, just completely by, well, actually, it is quite complicated to solve because you have to use the 45 degree constraint. Anyway, some readers found a really nice solution, another reader found a really beautiful solution for two Zambonis, so they would go together without colliding, that was also kind of important. So we will see, maybe Zamboni will turn to using it!

*You’ve also written about recent Russian immigrants. What is the story there?*

So, I do things that interest me, and I don’t sleep that much at night. So I like to write in the middle of the night.

*How much do you sleep at night?*

Well, like 5, 5 ½ hours, and then I take a few 10 minute naps during the day. So, I wake up in the middle of the night, and often I want to write, because I just like it. And sometimes, daily experiences influence me. So in the case of the Russians, I was director of undergraduate studies in the early 1990s. And a lot of Russians were coming to New York. They were kind of bimodal. Some of them were just brilliant. And some of them thought they were brilliant, but weren’t so brilliant. And there weren’t many in between for some reason. Well, one day, this young man comes into my office with his teacher, and he is a very young man, he is 12 years old. And his teacher says, “He’s in my freshman Pascal class, and there is a problem.” And I said, “oh, really, what is the problem?” Thinking, he was too young, he couldn’t get it. He said, no, Viktor is too good. I said, oh, really, so tell me Viktor, what have you done, what is the biggest program that you’ve done? He said, “Well, I’ve done a simulator for high temperature superconductors.” And I said, “Oh, that’s good.” And so my colleague asks him about the math he used and he was explaining Newton gradient methods and all those sort of things, and he certainly understood what he was doing. And so then I said, “Ok Viktor, which programming languages do you know?” “Pascal, FORTRAN, and Lisp, and C, and C++, and I think that is all,” he said. That was very good. “Do you know any graph theory?” “A little bit,” he said. “Could you write us a connected components algorithm?” “In which language,” he asked. So I said, “Oh, I don’t know, maybe a set oriented language.” So he does that in his 12 year old handwriting, from the top of the board to the bottom, without a cross-out, and it is perfect. So my colleague said, “Well, you have a loop there, and you have to compare two sets in that loop, that’s an expensive operation. What will you do?” “Well, I will probably use bit vectors,” he said. Then my colleague said you could also just have a flag that just changes if there is some change in the loop. And he said, “Oh yes, but it wouldn’t be so pretty.” So I felt that I had to write a book about these people. I wrote with a Russian playwright, and we wrote about mathematicians and scientists but also business people and sex workers and writers. So, it was quite an eclectic crew, and it was really a fun book to write. And I think what I try to do when I write is I try to write something nobody else is writing. I really respect people who write wonderful textbooks, like let’s say, a real database textbook, but I would never try to do that. Because I feel that already they are doing a great job. Even if conceivably I could do a better job, it would only be marginally better, and so, my books, for example, about database tuning, I wrote that because there was no such book. There were books about database tuning for particular systems, but never the general principles.

*Are there general principles? Can you generalize beyond what each separate system does?*

I think there are a lot of general principles. The main ones, I would say, are that people often tend to tune things that they see are bad even if they are not important. The other thing is that very often there are systematic problems because people don’t realize that there is a fundamental principle of all databases, in fact all of computing, which is that starting something is expensive, but once it’s going, it is quite cheap to have it continue. So for example, reading a sector of a block is almost as expensive as reading the whole block, and so I call that “starting is expensive, but running is cheap”. That occurs again and again, so if a person writes a loop in Java, and then emits an SQL query every time through that loop, that’s much worse than one SQL query and then going through the resulting data.

Another thing is that partitioning is not understood too well. So people understand partitioning data over different hardware, but they don’t really understand that there is also temporal partitioning. So very often, what you might want to do is, you might want to do some work during the day, and some work during the nighttime. And of course, when you say that, people say “well of course”, but it’s not the first thing that comes to people’s mind. And so for example, banks send out statements, when they used to send out paper statements, they would send them at 1/20 of the statements everyday. And that makes sense. It doesn’t matter that you don’t get your statement at the end of the month, it could be on the 12th, who cares as long as you get it every month? And similarly, for production databases, one can take advantage of temporal partitioning. So I think those are the principles, and surprisingly those principles are almost generative, you can apply them to almost every level of the database, from buffer management, to application design.

*So, is database, or at least the general principles of database tuning, is that something that you think would fit into a CS curriculum?*

Well, I think so, but of course I am very biased. I teach it, and students love it because it is very practical, and whereas, I think we do a very good job in this community teaching a database course that teaches the SQL and teaches normalization. Once we get to indexes and transaction processing, we have to recognize that most of our students are barely going to be conscious of that once they leave. Indexing perhaps, but not how to build the index, more what indexes do. So, although I think it is all very valuable, we should recognize, I think, that database tuning is what they will actually do, if they use databases, either as database administrators, or even as sophisticated application programmers. Because how many people are going to be working at Oracle or Microsoft and a few other places?

*Well, it is true that we tend to make our students take a course on operating systems, but most of them don’t go on to design operating systems. Does that represent a failure of our educational system, or is it okay?*

I think it is good, because, both operating systems and database internals give people a good sense of what system building is about, and that is important. And very often, people, students, later on write concurrent systems so they have to understand that. And operating systems, transaction processing are great ways to learn that. But, nevertheless, I think there is a place for database tuning, at least a little bit. I would welcome anybody in the community to take my notes, they are widely available. Philippe Bonnet has a wonderful infrastructure for a database tuning experiment, and they are also available.

*So the infrastructure, does that use a particular product, or is it generic?*

It uses mySQL, and then there are some scenarios that we have, we have some scenarios about a travel application, and some other scenarios.

*Tell us about your life as a fiction author.*

So, I always loved Sherlock Holmes. I really loved that. And I also loved puzzles because puzzles helped me overcome my problems in my first job. So, my first job was to design circuits for large mainframe processors at IBM. And although I had technically an engineering degree, really I’d only studied information theory. So, I get there, and I don’t really understand electricity, not really. And so, I bought myself a Heathkit and I built stuff, and I really tried to understand what was going on. But finally what really helped was reducing what I had to do to puzzles. And then through those puzzles, understand more details afterwards. So one of my jobs was to design circuits to check other circuits. And those circuits, the idea was that if a circuit failed, another circuit could say that circuit failed. But, circuit elements were still expensive enough at that time, that we didn’t what to completely duplicate the circuit. So we wanted something that was economical, yet still could check the other circuit. So if you take a decoder circuit for example, a 4-16 decoder, the output should have only one element that’s one, only one of the 16 lines should be as a one. So how do you do that in just a few gates? So that is one of the puzzles, in fact, that appears in the very first of my puzzle books. And the naïve way is you take every pair of the 16 and you see whether two of them are one, in which case there is a problem. But that is many, many, many circuits, at 16 choose 2 circuits, and one can do it much better if one looks at the 16 and says this is 4 bits, if 2 of these lines are one, then they must differ in one bit. And using that, you can design a much better and more efficient circuit. So puzzles always helped me.

And then I said, how am I going to give puzzles to other people? I’ll just give them informally to other people. And then I said, you know, I loved Sherlock Holmes, except for one thing, which is that Sherlock Holmes periodically would just leave and then come back and said, well I discovered something, and that something would often be the key to the whole mystery. And I felt that was unfair to the reader, the readers tried to solve this mystery himself/herself is robbed of that privilege because Sherlock Holmes pulled a wildcard out of his pocket, so I said no wildcards. So Dr. Ecco is kind of like Sherlock Holmes. He has a friend named Prof. Scarlett, and Prof. Scarlett acts the Watson role, and Prof. Scarlett is sitting there explaining the puzzle to us, but never does the reader get less information that Dr. Ecco. And so, if Dr. Ecco can solve it, the reader can solve it. And indeed, readers do solve it. Most of my puzzle books have puzzle contests where the solutions weren’t given in the back, and the person who won that contest got a free ticket to London for two, a free ticket to London and back from New York. So it has worked out well, it has developed quite a little community.

*So when you read a murder mystery, do you usually figure it out before the end who did it?*

Well, I try. Sometimes it doesn’t work. But I like to, I like that idea.

*So why is it that modern mysteries have to be murder mysteries, why can’t they just be mystery mysteries?*

I don’t know, I agree with you. I think there is plenty that is mysterious that doesn’t have to involve death. I mean, The Name of the Rose, for example, which Dr. Ecco’s name of course has two Cs, and Umberto Eco only has one C, but nevertheless, it was very inspiring book for me, because there was a mystery that fundamentally didn’t need to involve death, there was some death, but it didn’t need to involve death. And it really involved the reader, it really took the reader to an entirely different world. It was nice.

*What will your next book be about?*

Well, with Cathy Lazere, whom we mentioned before, we are writing now a book that’s at the publisher’s and it’s called Natural Computing: DNA, Quantum Bits, and the Future of Smart Machines. [The book came out in May 2010 in English and May 2011 in French and very soon in Japanese] And what it is, is a book that is much more speculative, but in the same format as Out of their Minds, which is that it’s biographies and then what people are doing. What we are looking at are future directions in computing based on people who are trying to solve extremely hard problems. So a typical hard problem is: how do you make spacecraft survive for a hundred years if they are going to make a trip? Another typical hard problem is protein folding. Another typical hard problem is robotics, and how do you make robots that fend for themselves. So there are many very difficult problems that require to me a new kind of computing and the new kind of computing we seem to be seeing is the computing based on a life analogy. Either they involve genetic algorithms, or they involve feedback in very interesting ways, so that systems are built to be adaptable. But genetic algorithms also involve feedback, so that is consistent with that. And very often, not only do they involve nature in a metaphorical way, but sometimes in a real way. So for example, one of the people that we interviewed does analog computing, which people thought was completely dead, but for certain things, it can really good. This summer in fact, we are doing an extremely intensive project with him at New York, his name is Jonathan Mills, and he is at Indiana. We are doing an extremely intensive project with him for protein folding, because, after all, proteins fold in a millisecond, the best computers for doing this, which is probably David Shaw’s Anton, take months to solve a small protein. And those are multi-million dollar machines! There is something wrong with the picture, because clearly, the protein is not computing like that. So it is quite possible that there are other paradigms that could work better.

*Beyond the puzzle aspects, are there other characteristics of the types of problems that you like to work on?*

What I am really interested in is finding vary challenging problems in any part of computer science. When we look at those problems, and look at how people are solving them, I think what we find is that there is going to be a lot more autonomy in computing systems. Even in sensor networks there has to be a lot more autonomy, because you can’t possibly go out and repair all these sensors. But it is even more so in some of the more wild things that people are doing, like in DNA computing, where there are literally billions of little DNAs running around annealing with another, or even in cellular computing where on your thumb you have more cells, little pieces of bacteria, than all the people who live on the planet. It is just an enormous question of scale. So I think what really are going to be interesting problems and problems where you say I don’t have control of the computing elements, and how can I nevertheless solve big problems. These problems will involve data in as much as each of these computing elements will have data. So, I think challenges like that will be entirely new and possibly give entirely fresh perspectives on all aspects of computer sciences including databases.

*Among all your past research, do you have a favorite piece of work?*

I have enjoyed so many things that I have done. What I really like, maybe, is that some of the work has surprising applications. So the tree matching work has had applications; we have gotten requests (because the software is available) from sociologists, from linguists, from all kinds of people. The time series work that we’ve done has been downloaded by the usual suspects on Wall Street, but also has been used by people who are interested in “query by humming”, which is one of our projects, and has been used in music, and also in astrophysics! So sometimes, the work has found a new life in unexpected ways. I think what I have really always enjoyed are large data puzzles, where there is a lot of data, but there is some combinatorial structure that you can make use of, and if that is the case, then I think that’s a problem for me. I don’t really like problems that don’t have a combinatorial structure, that are just large for large’s sake; I guess nobody does. But if there is a nice puzzle, if there is something clever that we can do, then I really like it.

*Do you have any words of advice for fledgling or mid-career database researchers?*

Yes, I would say that the advice I have is almost the same as the advice that Dijkstra gives when we interviewed him for Out of their Minds, which is, he said, “Choose problems that are at the limit of your ability, not too hard, but not too easy either.” Before you have tenure, you have to publish, but nevertheless, it is important to realize that you want to publish things that people will appreciate, that people will say “yeah, this was clever, and I couldn’t have done that”, or “if I could have done it, I would have been really proud of it”. The other thing is, to go out in industry or government, or wherever, and find people who are doing real problems. The founder of my institute, Richard Courant, when he came to America, he always would take his students to large engineering organizations and he would tell them to talk to the engineers. Not to the physicists and not to the mathematicians there, but to the engineers, because they would say this is our real problem. And I really believed in that because that has led me to database tuning, it’s led me to do the time series work, my work with the biologists lead me to write a funny little book called Statistics is Easy, and every time I have done something it is because something I have seen has motivated me to do it. Sometimes it is not research in the hardest sense, like Statistics is Easy is just an easy introduction to resampling statistics. But it is nevertheless I think useful because most computer scientists don’t like statistics, so I think utility is a great motivator, and I think it is sometimes underestimated in academia.

*If you magically had enough additional time at work to do one thing that you are not doing now, what would it be?*

I don’t know, I think I am doing everything I want to do.

*If you could change one thing about yourself as a computer scientist, what would it be?*

That’s a good question. I think what I would change is that I am curious about many things, and sometimes that leads me to try to do too much. So, and sometimes, therefore, I can’t do some really great work in that area. As I said, I am working with biologists, I am doing a DNA computing project, I think it is great for students, because all the students gain a lot from it, but sometimes it doesn’t end in, let’s say, a publication. I’m not really too worried about it, but I think that would be the one thing I would change, I would maybe say no more.

*Well, how do the students get a job, because you were saying you have to publish if you don’t have tenure, but of course afterwards when you do have tenure, your students will be looking for jobs, and if they don’t have the publications they can’t get that job.*

I quickly evaluate whether a student is likely to go to academia or not. Many students don’t really want to. If the student does, then I concentrate more on publications. Of course, sometimes I am wrong. I had a wonderful student named Yunyue Zhu, he won a best paper award, beautiful publication record, and he did beautiful work in time series, and then he went off to Wall Street because the money was so good! So what can I do?

*Is he still on Wall Street?*

Yeah! The whole thing about Wall Street is that you see the headlines, and you think the world is coming to an end, there were a lot of people who did extremely well, and all my students there are completely employed. It is a very funny world between the headlines and what’s going on.

*Thank you very much for talking with me today!*

It’s a pleasure, thank you so much, Marianne.