Jeffrey Naughton Speaks Out

on Database Systems as Control Freaks, How to Choose Students, How to Choose Problems, How to Get Attention, the Importance of Being True to Yourself, and More

by Marianne Winslett

Jeff Naughton
http://www.cs.wisc.edu/~naughton/naughton.html

Welcome to this installment of ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we're in the Department of Computer Science at the University of Illinois at Urbana-Champaign. I have here with me Jeffrey Naughton, who is a professor of computer science at the University of Wisconsin at Madison. Jeff's research has focused on improving the performance and functionality of database management systems. He received the National Science Foundation's Presidential Young Investigator Award, and he is an ACM Fellow. Jeff's PhD is from Stanford University. So, Jeff, welcome!

Thank you.

Jeff, you've gone on record as saying that database people are control freaks who don't play well with others. Do we also run with scissors?

[Laughs.] I don't know about running with scissors. I think you're referring to the title of a talk I've given in a couple of places, about database systems being control freaks. I was implying that the *systems* are control freaks, rather than the people who build them. Although, I guess you could argue that the systems reflect their builders.

Time will tell whether the title is just a gimmick to get a catchy title for a talk, or an insight into something. The talk title comes from a feeling that has been growing as I talk to people and ask them, "You've got all this data, you're trying to manage all this data; why aren't you using a database system?" The kinds of answers that come back are very often things like, "It's just so painful and unpleasant to use a database system. I have to conform to a relational schema if I'm going to use a database system. I don't think of my data in terms of a relational schema, but I have to define a schema to describe everything about all the data I want to query. And I have to do all this up front, [in advance]. Then I have to load the data into the database system before the system will manage it and help me run applications over this data. And once I've loaded the data into the database system, then pretty much the only way to get at the data is through the SQL query interface, and I don't really want to write SQL queries." So this is what I mean by a database system not playing well with others. It just is asking too much that if you want to use a

database system, you have to come to the database system and you have to accept everything that the database system dictates. It's like a toddler that says, "I will play with another kid in daycare, but I will only play with them if they play the game I want to play, in the way I want to play it, when I want to play it."

I can mention a couple of other issues in the scientific computing community that I work with. They run on a lot of different platforms, most of them parallel. They have to be able to have that same data on all the different platforms. That means the database management system would have to run on all these bizarre parallel computers, with new parallel architectures popping up every few years. The DBMS would also need to be cheap, and parallel DBMSes are not cheap things to buy, or to take care of, either..

So what's the solution to these problems?

I'm kind of hoping that the problems are a full employment act for database researchers to work on trying to develop technology that can solve these problems. I don't think there's a simple, easy fix to the problems. I think that maybe part of the answer is that we database researchers could try to adopt more of a mentality of the kind you were suggesting [in an earlier conversation]: let's look at the problems facing these people, and try to figure out whether there is any way to provide some of the benefits of a database system without falling into the same kind of old megalomaniacal control over everything that goes on with the data.

> Successful students seem to be [able to] express to others why it is that they're interested in working on [their chosen topic], rather than just viewing it like a course assignment.

There's a danger in that suggestion for young people, not for you, but for young people---

I'm not young?

*I mean **young** young, like pre-tenure; infants in the research community. If they go off and do something that's completely new, in the past the database community has not been exactly welcoming of things that strike off in a radically different direction. And the classic review for a conference paper about such research says, "This is not database research."*

Right.

So, are you speaking to the young people? How could they overcome that problem? Or are you speaking to the rest of us who don't have anything left to lose?

[Laughs.] Excellent question. I mean, I would love to see the young people feel comfortable to go off pursuing these new kinds of things. It's a very difficult problem. I've actually tried to think about where this problem comes from. One of the things that I find is that the closer you go to something that resembles what the commercial people live with every day, the more likely you are to get really hammered in your reviews, because the commercial people have thought about all these problems and they know some twist on it that an academic starting out might not know. This means that it can very hard to come up with a fundamental core database technology kind of paper that will be acceptable to such a reviewer, because they'll look at it and without naming any specific systems, they'll say, "This could never work in our system, because our customers want

this and that." So you can't go too close to the core traditional database technologies. It's hard to get things published in that domain now, but what you say is also true. If you go too far away, then people will look at your paper and say, "Well, maybe that's interesting, but it's not database research." And you have to somehow try to find a happy medium there, and I guess I don't really know what the boundary is of database research now. What should it be? That's a very good question.

The secret to closing in on Mike [Carey's record number of SIGMOD papers] is to have Mike Carey slow down.

That ties into several things I was hoping we'd talk about today. One of those has to do with sending papers to SIGMOD. You're fast closing in on Mike Carey's record of coauthoring the most papers at SIGMOD. He's at 28 papers, you're at 24, as of today. So can you give the audience additional tips on getting papers accepted to SIGMOD?

The secret to closing in on Mike Carey is to have Mike Carey slow down. That's very clear, and I can tell everyone how to do that. But in terms of getting a lot of papers accepted... I've been really fortunate to have such an extremely wonderful group of students and colleagues and collaborators and just a tremendous environment up at Wisconsin to work in. I don't know if it's anything special I did. I was in a wonderful place with wonderful people at the right time.

Any tips on choosing students?

You ask good questions. I don't have any secret tips for choosing students. It's almost something I can't put into words. With a certain student, you'll get the feeling that you can work with this person, and you do. It's not any abstract metric that you can list, such as GRE scores above this threshold, GPA above that threshold, or anything like that. One of the things that is really nice is if you can find people who are storytellers. I don't mean people who tell jokes, although that's fun. I mean people who can tell the story about what they're working on and why they're working on it.

Does that mean they're young people who see the big picture?

Perhaps, or even are just able to explain the small picture that motivates them to work on what they're doing. Given a specific paper that they are working on, it's the ability to realize why it is that they are interested in writing this paper. Maybe you could say, "I'm being very mercenary; I need to write this paper so I can get a PhD and get a job, and then I need to write more papers so I can get tenure." But at some point you run out of these kinds of reasons, and then you ask, "Why am I writing these papers?" And one justification that I think we all fall back on is that something in the paper is interesting, that caught your attention. Successful students seem to be the kind who are able to capture that and express to others why it is that they're interested in working on this topic, rather than just viewing it like a course assignment.

A harsh criticism of the database research community as a whole would be that people work on a problem like a drunk looking for their keys under the lamppost: because that's where the light is. They see a problem. They know how to solve it, or they think they know how to solve it, and that's why they're working on that problem. So, do the students you choose have a broader motivation, or is it that they see a problem that they have the tools to attack?

I see some of both. We're getting near an even more difficult and bigger question, which is how do you decide what's valuable work and what work warrants doing. The fact that you have the skills to solve a problem and can solve it doesn't mean that the problem is interesting. It also doesn't mean that it's *not* interesting. So I'm getting a little tangled up here but the....

Sometimes I almost get the impression that people think that in the database community, we're all sitting there saying, “Look at that beautiful sea change problem over there. Wow, that's a big, wonderful, great problem, but I don't want to work on that. I want to work on this small little incremental thing instead.” I don't think that's what's happening.

Let me pull you out of the tangle by asking where you think database systems research should be going.

I don't have a definite answer to that. I've got some reactions to certain criticisms that I've been hearing of the database community, criticisms from within the community. And by the way, all this criticism is good. I think it's fine that we're criticizing ourselves.

I think we’re being unfair to ourselves with certain lines of reasoning, like the criticisms that nobody's looking for the big picture anymore. You know, “All you folks are out there polishing a round ball, and not working on a big sea change. For a while, there were all these talks where almost every speaker would use the phrase “sea change”: “We need some sea-change problem to work on, something that will really change the way we do data management, something that will have a big impact.” And you can't argue with that. I mean, how can you say, “No! We shouldn't be working on stuff that's a sea change and will change the way we manage data”? The interesting question is, if we aren't working on those kinds of problems, why aren't we working on them? Sometimes I almost get the impression that people think that in the database community, we're all sitting there saying, “Look at that beautiful sea change problem over there. Wow, that's a big, wonderful, great problem, but I don't want to work on that. I want to work on this small little incremental thing instead.” I don't think that's what's happening. I think we're all desperate to find a sea change thing to work on. This is reflected in the way the research focus of the community changes. I think that the community is in many ways rather light on its feet. Just look at all the different things that have happened, even in the course of your and my careers. There are logic databases, object databases, object-relational databases, XML databases, stream databases, and it just keeps going on and on. I think this is a sign of a healthy community.

But what is it that the database research community should be doing? To answer that question, you have to ask, what is the role of a database researcher? Many times I think we all get into trouble if we accept definitions of who we are that don't really fit who we are. So let me give you some examples of definitions that I don't think are good. One definition is, “The database research community is an extension of the development organizations of the commercial database companies.” If you accept that definition, then there's a very clear metric for evaluating our research: does it have impact on a product? Can you say, “Look at this line in that commercial system, that line of code right there, that was influenced by my research, my idea.” I don't mean to denigrate that metric, because it’s wonderful when that kind of influence happens; but not everyone is called to be the kind of researcher who sees themselves as an extension of the

development groups of commercial companies. If you pick someone who is not really working in that mode, and evaluate them as if they are working in that mode, then they will fall apart.

I don't think the whole community is an extension of the development organizations. I think that at some point, many of us do feel that way, but at other points we don't. As a professor in a university, as I get older---you're already said I'm old---but as I get even older… I should point out that we graduated at about the same time!

Well, of course. I don't interview anyone younger than me.

[Laughs.] Okay.

You just barely made the cutoff.

Just barely made it in. So as I get older, I think that the main product or contribution or impact of most academic people is their students. Some of them may have been students in courses that you taught, and you may not even know that you've had an impact on them, although you hope that you've intrigued some of them enough to start them thinking about database issues and pursue this kind of a field. And that's a very strong impact that's hard to measure on a resume.

I think I got on this topic when you asked me what the database research community should be doing. The reason I have a hard time answering that is that I think that there are so many ways that we can make contributions. You can have the idea or the system that changes the way data management occurs. That's great. Or you can come up with an undergraduate course that starts training people and getting them interested in pursuing data management as something they want to work on for their whole life. The database community is doing a mix of all those things, so I think we should call it successful.

I've moved to the point where neither … the theoreticians or the hard-core systems people would think of me as one of theirs.

You've worked on relational databases and object-oriented databases and object-relational databases, and now XML databases. We could view you as a data model opportunist, or perhaps more realistically, as a data model agnostic. So what's next? XML-relational databases?

[Laughs.] I don't know what's next. Perhaps I am a bit of an opportunist. I will go back to saying that one of the things I think we need to do as database researchers is to explore things that we are not sure will be successful. We shouldn't explore things we are *sure* won't be successful---we've got to avoid that. But when an idea for a new data model comes along, it's rather difficult for a big company to justify spending research dollars on it, because they have so much invested in the old model. So when a new data model is worth exploring, we should see how far it can go. It's unfair to say in hindsight, "that data model didn't go anywhere, so we shouldn't have ever investigated it," because I think we're going to need a fair number of misses before we have another hit.

Is XML a miss?

XML is interesting. The critique of XML as a data model (as most of the people reading or watching this will know) seems to go, "You stupid young people," ---and I guess they call me a

stupid young person in this context--- “you have not learned the lessons of your elders. You’re working on this tree-structured data model, and we tried that in the late 60's and early 70's, and it was awful. But sure, go ahead and try it. You'll hate it too.” But my view of XML and how we're working on it is that we're not saying, “I have a new idea for how we should store data. We should store it as XML. Let's create everything as XML. That’s how Walmart should do their data warehouse, as XML.” XML is here to stay. There's gobs and gobs of XML being created and generated and stored, and people want to manage it and query it and handle it somehow. The reason why I got attracted to XML was not so much that I thought it was a new wonderful way of handling data, but that there was a new wonderful kind of data that we had to handle. So I don't think that XML will be the next big data model. If I had to get out my crystal ball here, I would say that the future looks pretty relational to me. Or maybe object-relational, with support for many more different kinds of object types. Much better integration with text would be a huge change. If I had to pick one thing that would be changing over the next ten years, that might be it.

Your comments on XML being something that's here to stay so we should just deal with it tied back to your earlier remarks about us not polishing a round ball. You said that we should go out there and see what needs to be done and do it because it needs to be done, not because it fits with the old way of doing things.

Right, yes.

At the time I came to Wisconsin, apparently every assistant professor was assigned a full professor as a programmer. So I got David DeWitt as my programmer.

During the course of your career you have moved from pure theory to hard-core systems. What led you to make this change?

I don't know if I've made it to hard-core systems. I would say that I've moved to the point where neither community would claim me now: I don't think the theoreticians or the hard-core systems people would think of me as one of theirs. I'm sort of in the middle. What led me to do that? While I was a graduate student I led a double life. (When I tell people this very few of them believe it.) I did very theoretical database research at Stanford, and then at night and on the weekends, I would go hack at DEC’s Western Research Lab. They had this new ECL processor called the Titan, and we were working on porting Unix to that. I've always been intrigued by hard-core system building and I've always enjoyed theory also. So certainly my publications started out very theoretical and maybe I picked up that accent at Stanford, from the people I was around. My advisor, Jeff Ullman, provided a wonderful environment for exploring theoretical issues related to databases. And then when I finally landed at Wisconsin, there was this incredible hard-core database systems group. People may not realize this, but at the time I came to Wisconsin, apparently every assistant professor was assigned a full professor as a programmer. So I got David DeWitt as my programmer. He had just built this system that he and his students called Gamma, a tremendous parallel database system. And we would get coffee in the morning and talk about new algorithms and new ways to do things, new problems. We fell into this mode where I would think of some crazy algorithms and he would throw out most of them, but he would start implementing some of them, and we started working together that way. It's been tremendous, it's worked out really well.

Someone suggested that I ask you, and I quote, “whether PODS publishes anything worth reading nowadays.” I think I'd rather ask you what you think database theoreticians should be working on today.

Wow! It's very tricky to tell a theoretician what to work on. If I'm being completely honest, I think theoreticians should work on whatever interests them, because I have this belief that we human beings are wired to find important things interesting. Somehow this seems to work. When you look back over time, some very interesting theoretical work has been done just because the person doing it found it fascinating. The old stock example is probably number theory. The person who developed number theory had some quote, "I do it because it has no possible application." And now we find that cryptography is making heavy use of number theory. Another example is the people who did mathematical logic. They certainly weren't thinking of it as a framework to help you manage large data sets in a computer. They were doing it because they thought it was interesting. So I would not like to see theoreticians work on little corners of problems they're not very interested in but that might have some application. If you're going to be a theoretician, remain true to it and find elegant, beautiful stuff that's inspired by some practical application, but do it mostly because you find some beauty in it.

What's your most interesting consulting experience?

If I'm being completely honest, I think theoreticians should work on whatever interests them, because I have this belief that we human beings are wired to find important things interesting.

I guess this story is about a consulting experience. I'm not sure, because it started out when there was some funding from DEC, back when there was a DEC. David DeWitt, Mike Carey, and I started working on the OO7 Benchmark. Rick Cattell had done OO1, Object Operations 1. Mike Carey suggested we skip two through six and do OO7. So we did. And it was just an amazing experience for me because it was the first time that industry and the popular trade press had shown any interest in the kinds of things we were working on. It was a strange experience. We did this benchmark, we got three or four of the object-oriented database companies roped in, and we started running tests on all of their database systems. Somehow it got out that we were doing this, and I started getting calls from trade publications. Editors would call up and they'd say, "Who won?" And I'd try to say, "We don't rank them. We don't actually have a winner here. We just test many different aspects of the systems and people are supposed to look at that and figure stuff out." But they kept pushing me. I remember at one point I was invited to go out to Stanford to give a talk at the database seminar on OO7. I got to the room just before the talk, and I'd never been in a room that full. It wasn't a very big room, but it was so full that when I was giving the talk, I couldn't walk. I couldn't take a step. I had to stand in one place and give the talk. And in the back were all these video cameras that people had brought in to tape the talk. That was an amazing experience. My advice to a young researcher starting out is that if you want a lot of attention, what you should do is find some kind of benchmark and benchmark commercial systems and publish the results. The companies will all sue you, but you'll get a lot of attention.

Your first faculty position was at Princeton, but you soon left for Wisconsin. What led you to make the shift?

It was really entirely for personal reasons that I made that shift. I finished up at Stanford, went out to Princeton, and just really loved it there. It was a great place. There were several people I was working with there, primarily Kai Li, who was not really a database person, and Dick Lipton, who was an everything person, and Hector Garcia-Molina, who was just wonderful to have as a mentor. Everything was great. And then what happened was that there's a course at Princeton---I think it was either 103 or 109---Introduction to Programming. The rule is that if you teach that course, either you or your spouse has a baby, or you are Ken Steiglitz. That was the invariant,

because everyone was teaching this course and having kids. And I made a mistake. I taught it back-to-back two semesters, and so we had a set of twins. I'm from Madison and my wife's from Madison, so we went back one winter break, and DeWitt is a very shrewd man. He gave me an office for the two-week break. And I'd go down and work all day and when I would come home at night, the babies seemed happy, we were all happy. It seemed like a great place to be with all our families. It seemed to be a wonderful place to be personally, and it was professionally obviously a great place to be, and at the end of two weeks DeWitt said, "Why don't you move here?". I went home and said, "Why don't we move here?" That was about it. So it really had nothing to do with professional considerations.

I'm sure there are lots of young faculty members out there with children facing similar pressures.

Yeah, that's tough. Kids come at a hard time in our lives.

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

Perhaps this is too philosophical to be useful, but it's very important to realize what kind of a researcher you think you are best suited to be, and follow it. Don't look at some superstar and say, "I want to be like her, and she has a big group building lots of systems, so that's what I'm going to do." If that's what your calling is, that's wonderful, go do it, more power to you. But don't get confused by trying to emulate someone who doesn't fit your working pattern.

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

I've got this feeling that there's a tremendous body of work out there that I don't know enough about. There are all these young people pursuing interesting things. I run into them at conferences, I run into them at places like here [at UIUC]. I hear about things that I just didn't know about, and I wish I could spend more time keeping up and knowing what they're doing.

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

Up till now, things have gone as well as I could possibly imagine for me, so I'm just extremely happy with the wonderful people I've had a chance to work with and the experience I've been able to have. So it would seem ungrateful to say I want to change something about it. I think I'm pretty happy with it.

Okay. Very good. Thank you for coming.

Thank you.