David DeWitt Speaks Out

on Rethinking the CS Curriculum, Why the Database Community Should be Proud, Why Query Optimization Doesn't Work, How Supercomputing Funding Is Sometimes Very Poorly Spent, How He's Not a Good Coder and Isn't Smart Enough to do DB Theory, and More

by Marianne Winslett



David DeWitt, http://www.cs.wisc.edu/~dewitt/

This issue's interview with David DeWitt took place in November 2001, in Roy Campbell's HDTV studio at the University of Illinois at Urbana-Champaign. Once again I'd like to thank everyone who suggested questions for this interview and for the other interviews in the series. As usual, all errors of transcription are my own and the videotape itself is the final arbiter of meaning. This interview was recorded digitally, and we expect that the video will eventually be made available on the SIGMOD Web site.

In an upcoming column, we'll be hearing from Hector Garcia-Molina.

==========

I'm Marianne Winslett, and I'd like to welcome you to this installment of the SIGMOD Record's Distinguished Profiles in Databases interview series. Our guest today is David DeWitt, who is the John P. Morgridge Professor and chair of the Department of Computer Science at the University of Wisconsin at Madison. He is a member of the National Academy of Engineering; he is an ACM Fellow; [he is the local arrangements chair for the SIGMOD/PODS conference this year;] and he is known for his work on performanc eevaluation and parallel databases. So, welcome, DeWitt!

I'd like to start with some questions about life in academic departments. You've had a career in a field that is closely related to industry, in a place where there isn't any database industry. Would you advise fledgling database researchers to go to a place that is a hotbed of industrial activity, or doesn't it matter?

I don't think it matters too much. I think the main thing is pick a department that is supportive and willing to build a group. I think you can build a strong group in any university that is supportive. I think we [at Wisconsin] have shown that you can do databases in the middle of the

country fairly effectively. I think there are other strong groups at places other than [the east and west US] coasts.

[In systems research,] sometimes you get good software artifacts, and sometimes you get lots of papers, and occasionally you get both

Speaking of building a database group: most academic departments have one or two people doing database research, but at Wisconsin, for many years, you had five or more people doing that kind of research. As the numbers increase, is there a qualitative change in what happens, or is it just more of the same?

I think there are definitely advantages to having groups of four or five faculty members in an area. If you look at Wisconsin's department, we've tried to organize all groups as four to five faculty members. I think [having just] two people [in an area] presents a problem in that they may go through periods where they don't want to work with each other, [though of course] sometimes they do want to work with each other. But when all five of us were [in Wisconsin's CS Department], Mike [Carey] and Jeff [Naughton] and I and Raghu [Ramakrishnan] and Yannis [Ionnides] plus Miron [Livny] (and Tony Klug before any of these other people were at Wisconsin), there are all sorts of permutations [of working together] that happened over time. So five is a fun number [of faculty to have], and it's much better than two. [In sum,] I think there is a qualitative difference in addition to a quantitative difference.

You've led large software projects in academia, which isn't all that usual: lots of money, lots of people. How do you divide your efforts in that kind of work between producing papers and producing software artifacts?

I think [the way work gets divided between papers and software artifacts] is a total accident. I think back on my two most recent projects, the Paradise project and the Niagara project. The Paradise project---if you look at the number of papers per dollar spent, it was [very bad]. We produced a great software artifact, had a great time doing it; at some point there were actually 25 people working on the effort, between graduate students and full-time staff members. It was far too big a project to do in academia. [However], the number of papers produced was [very low], and the dollars per paper was really high. Whereas in the Niagara project, we had a [difficult] time producing a [reliable] software artifact, [but] we [have produced] lots of papers. I think you can't plan it; you just go with the flow, and sometimes you get good software artifacts, and sometimes you get lots of papers, and occasionally you get both, [as happened in] the Gamma project. But [the Gamma project] was the exception [in that regard].

Which is more influential in that kind of project, the papers you produce or the software that comes out of it?

Depends [on] what you are trying to sell.

Well, in those three projects, what were you trying to sell?

I'm not a very good coder, and the software artifact, by the time we were done with it, was unusable

In the Gamma project, we were trying to show proof of concept. Eventually Naughton and I were the last

programmers working on the software artifact. Jeff designed the experiments and I wrote the code. I'm not a very good coder, and the software artifact, by the time we were done with it, was unusable.

> [For] a lot of people, programming is boring … [It's not] that big a part of computer science any more.

But it proved the concept?

It proved the concept. [Whether the project's papers or software artifacts are more influential] depends on the students [who work on the project] and the skills of the students. Sometimes you have good ideas that turn into good software artifacts, but sometimes the ideas are bad ideas. I think you need to exploit what you currently have in terms of the mix and quality of the students---their software skills and their research skills.

Shifting gears slightly, what should go into an introductory computer science class for majors?

> [The Wisconsin Benchmarks] made a lot of people, including a lot of friends, very angry. … Larry Ellison was very, very angry and---I guess this is the best story--- tried to get me fired.

Boy, that's a good question[,] a loaded question. We're actually looking at this [question], because if you look at the number of women (at least at Wisconsin) in the introductory [CS] classes, it's very low. And the question is, why is it so low?

How low is it?

I think we're about 22% [women] in the introductory class, but the number of majors drops to 10-15%.

[My own department is] lower than that.

And the question is, why? I don't know why. I have two daughters, one of whom is a chemistry major and one of whom is a math major. [… Neither daughter] has ever taken a computer science class[, although] the math major is going to be required to take a computer science class. But there's something about young women in high school that [causes them not to take] any computer science classes, despite being perfectly capable of [doing computer] science. I don't know why that is; I don't know whether it's because [computer science is] viewed as [being] too male-dominated, or whether it's just viewed as boring. I think part of the problem is that we teach programming first. And I think for a lot of people, programming is boring, and it's not a good reflection of what goes on in computer science. If you think about chemistry, chemistry starts by teaching inorganic chemistry, and quantitative analysis is a very small part of the introductory course; [the introductory course is] a broad brush of inorganic chemistry. Our introductory courses could have some architecture in them, some theory in them, maybe some database systems in them. But they don't need to just be data structures and programming.

Would they be hands-on [courses,] then?

Maybe, maybe not. I don't think programming is that big a part of computer science any more. I think there are a lot of things that you can do in computer science that don't require a lot of

programming skills. I think we should just try something different and see whether that affects the number of women in the field. [Of course,] it may not have any effect.

Have you tried [this new introductory course approach] at Wisconsin?

No, but we have a curriculum committee that's made up of all junior faculty members who are just fresh out of their PhD's, with no senior people. We haven't changed our courses in basically 25 years. The introductory sequence looks exactly the same, in terms of what gets taught in what courses. So I've tried to get new junior faculty members to think a little bit differently, to try something different. We're going to try to do something completely different in the introductory course with a very minimal amount of programming that goes on in the course.

We haven't changed our [intro CS] courses in basically 25 years.

Interesting. I look forward to hearing how it turns out.

Actually, I wish someone else would do it so we could just copy their course.

You can be the trailblazer for the rest of us.

Many years ago you were one of the authors of one of the first popular database benchmarks, the Wisconsin benchmarks. Are there any stories that you would like to tell about the Wisconsin benchmarks?

Not on videotape!

Actually, it was an interesting experience. It got a lot of people's attention. It made a lot of people, including a lot of friends, very angry. I remember Mike Stonebraker once getting really mad at me because we had shown that Ingres didn't do very well on a particular kind of query. I think a lot of people got really emotionally caught up in the performance results instead of trying to take the results and use them for what they were intended for, which was to say, here's where your system works and doesn't work.

There *is* the case where Larry Ellison was very, very angry and---I guess this is the best story---tried to get me fired. [He] didn't quite understand the concept of tenure, didn't understand the concept that the department chair wasn't going to fire me because I didn't say very positive things about Oracle. But I think all in all, benchmarks have served the community well. I think that they help the developers focus. In general, I think the whole benchmarking effort has been very positive for the community.

Are you suggesting that professors shouldn't work on benchmarks unless they have tenure?

(Laughs.) Yes, I'm definitely suggesting that! The sad thing is that every database product [except DB2, I believe,] has a clause in it that basically is the result of the Wisconsin benchmarks, saying no one but vendors can publish numbers. I think that's really sad. I think that's a silly attitude for the industrial community to have. If you sell a product, people should be able to evaluate that product. The database [vendors] seem to have some kind of phobia about people evaluating their products.

But [the vendors' benchmark] results that they publish are independently audited, usually.

No, they're never audited any more. There are some rules that [the vendors] have to follow in reporting their benchmark numbers, but I think it's widely agreed that no customers ever do as well as the vendors do.

Well, that would be true: a benchmark [result published by a vendor] is a guarantee that your performance will never exceed the published number.

That's right, that's certainly the upper bound.

I'm not sure why [NSF's] CISE has an advisory board, because I think our advice is repeatedly ignored.

I think this restriction has allowed vendors to concentrate on one particular number, whether it be TPC-A or -B or -C or -D or -H, and it has hurt the community in general, or users in general, because users can't conduct their own evaluations and publish their own evaluations. That allows the vendors to focus all their effort on a single number, and I think that's the wrong thing.

Well, you can publish as long as you call it database [system] A, B, ---

--- C, or D. Yes, that is the standard dodge for getting around it, but [still!]

I think there's been a lot of bad money spent in the name of supercomputers.

Has startup fever been a good thing or a bad thing for academia?

I think it's been bad in that some of the very best students haven't stayed to get PhDs. It's been good in that some academics have done very well financially. I think in general it's been neutral. It has hurt the PhD quality, I'd say.

What about now that the fever is winding down?

I think it's great. Everyone wants to stay and get a PhD now. I think there will be a swing back and students will be more conservative about leaving academia after the Master's degree. I think it'll be good for academia for a couple of years.

What about the recent economic downturn in the US? What do you think will happen in academia because of that?

I think the same sort of thing [will happen]. I think that there will be more applications for grad school. I think incoming graduate students will be better. I think they will stay longer. I think that means we will produce more high-quality PhDs and hopefully [also] more students interested in continuing in academia.

What about research funding, though? That pays for those students.

I think the real question is, will the government, after September 11, be able to afford to fund everything that they need to fund? And will there be a trickle down effect on more basic kinds of research? I think that if you're in security---this is a great time to be in security. If you're in database systems, it might be a pretty good time because they're going to have to manage a lot of

information. The question is simply will the government be able to afford [to fund all that needed work], and I don't have any idea [whether they will be able to do it]. Database systems, and information management, will become increasingly more important as the government tries to collect more information. And there are obviously privacy issues that we have to worry about. I think that it could be a good time [with respect to funding] for the database community.

Continuing with the questions about database funding, I know that you are a member of the advisory committee for [the US National Science Foundation (NSF)]'s CISE, and [CISE] is the source of most of NSF's funding for database research, as well as research in many other areas. Do you think that NSF should be funding people, or funding specific research projects?

I think that they should be funding more things [than they do now]. It's good to sometimes fund people. Sometimes proposals are a little too narrow. But I think you need to be able to fund new faculty members, so sometimes you need to fund proposals. But funding people is also perfectly viable.

I don't think the CISE advisory board has much impact on what CISE does, so people shouldn't think that I have a lot of say in who is getting funding.

What do you advise them about, then?

No matter what we say to them, they never listen to our advice, so it doesn't matter. I'm not sure why CISE has an advisory board, because I think our advice is repeatedly ignored.

I think the whole funding situation, even with [NSF's] ITRs, is pretty discouraging. … [I]t's almost impossible to get money to do core [database] research

You said that you thought that [NSF CISE] should fund more projects, but you also said that that you thought some of the project proposals were quite narrow.

If you say that you want to work on X, and X is really broad, I think it's harder to get that kind of project funded. A typical strategy [to get a grant] is to do the research, and then write a proposal that proposes to do the research---and I think that's unfortunate. I think people should be able to say, I want to work in this broader area; and that may be what I mean by funding people.

I think the whole funding situation, even with [NSF's] ITRs, is pretty discouraging. In the old days, there was a program called Coordinated Experimental Research, CER; those grants were started in the late 70s and they ran about a million dollars a year, and you could really do significant software development [with that amount of money]. Nowadays, the biggest ITR grants you can get are in the range of a million dollars a year---and twenty years have gone by! You get a lot less out of your million dollars a year [now] than you could in the old days. I think it's really unfortunate. Personally, I think CISE puts far too much money into supercomputers and terascale and grid computing and all the things that [the University of Illinois at Urbana-Champaign (UIUC)] gets.

[It's true that] we're a hotbed of that [kind of funding and research].

I think [that that kind of funding is] not funding computer science; it's funding the physicists, and it's not funding computer scientists.

Well, it's funding me. *That and my security work, the topics that you brought up...*

Well, that's ok, that's fine. I think [that that kind of funding has been given] too much money. I think that building 2000-node clusters and claiming it's computer science is nonsense.

Don't you want them to be able to simulate the nuclear arsenal, though, [instead of conducting above-ground tests] and all that?

I think that's funding physicists, it's not funding computer science research.

Oh. Well, I think they need a lot of help in order simulate the nuclear arsenal, it's pretty tough [to write that kind of simulation].

I think there's been a lot of bad money spent in the name of supercomputers. I think [the] PACI [program] was a perfect example of money that was not well spent.

> I think we [database researchers] have done an absolutely terrific job as a field and everybody should be proud of that.

to

Ooooh, hitting home at Illinois! [(PACI is a major source of funding for the National Center for Supercomputing Applications (NCSA) at UIUC.)]

You're the interviewer, [you asked the question, and] that's how I feel.

No, that's fine, that's fine. Are there specific things you wish that PACI had produced that didn't [happen], or do you just think that whole direction…

I think it's perfectly fine to fund big pieces of iron [(i.e., supercomputers)], [because] you need to have national centers like [NCSA at] Illinois, and Pittsburgh [Supercomputing Center] even, and San Diego [Supercomputing Center]; I think you need to have national centers where people can do computation outside of the [big supercomputers at the] government labs. But I don't think you should tie funding iron with funding research. That's my problem with the PACI program. I think that it tried to tie funding iron with funding research and funding applications, and I think they should be separated apart. I actually like the Pittsburgh [Supercomputing Center's] model better, of just funding that iron and funding the research separately, [rather] than funding everything as one big lump sum---because I think there's more accountability [that way].

I was about to ask why, but you told me already: more accountability.

[More accountability] to the funding agency.

So you mean that, for example, if they were successful in building the iron, they might call the whole project a success, even if …

> I think too many people get into hot areas. … Jim [Gray] wrote this really nice paper on data cubes[, and] all of a sudden we had three hundred people writing papers on data cubes.

I don't think there's much research in building the iron. [You just need to] buy a bunch of machines, put them in a machine room, cluster them together, attach them to the grid. I just think

that purchasing the hardware should be [the extent of the use of the funding]. Obviously if you purchase hardware, you should [also] support that hardware; but you shouldn't necessarily have the people who get the grant [also be the people who] determine which research projects should be funded. I just don't like that model, which is why I dropped out of the PACI program.

[Q]uery optimizers [do] a terrible job of producing reliable, good plans [for complex queries] without a lot of hand tuning. I think we need to totally rethink how we do query optimization

I see, interesting.

I don't think the SIGMOD community is going to be interested in that.

Well, I'm *interested in it. That's the world I live in. [If it seems to be too boring for the SIGMOD community,] we can just trim it out of the printed version of the interview.*

The traditional core areas of database research aren't as well funded as they once were. Is this just a sign of the maturing of our field, or have we missed some core areas that need more research?

I think we have missed some core areas that need more research.

[Optimizers] are making assumptions about joins five or six levels up in the tree based on just wishful thinking.

[First, let me say that] I think the field certainly has matured. We now have very, very capable systems, and the field should be proud of that accomplishment. I think both the academic researchers and the industrial people have done a terrific job. The systems are reliable, they're scalable, [they provide] high performance. I think we've done an absolutely terrific job as a field and everybody should be proud of that.

[However,] I think there are some core areas [that need more attention.] I think query optimization is a huge hole; I think I/O is a huge hole. I think too many people get into hot areas. It was recursive query processing for a while, then it was object-oriented databases, then it was data cubes; because Jim [Gray] wrote this really nice paper on data cubes[,] all of a sudden we had three hundred people writing papers on data cubes. Now we have data mining, where [the Knowledge Discovery from Data (KDD) conference] has seven hundred attendees. I think people get hooked into fad areas---which is fine, because I think that leaves a small group of us interested in core problems.

There's been very little funding for core [database] research. [The US Defense Advanced Research Projects Agency (DARPA)] hasn't been interested in that for years and years; DARPA has nothing going on in databases right now, although that may change, and NSF isn't interested either, so it's almost impossible to get money to do core research.

You said [that] query optimization [needed more research]. So what part of query optimization do people need to do more work on?

All of it! Query optimization is 22 years old at this point. Everybody does exactly the same thing, all based on work that Pat Selinger and the System R team did, and it doesn't work [any more]. [Database] systems have gotten ever more powerful. [Now we database systems users] can do ten-way joins, we can do TPC-H queries (which are incredibly complicated queries) on huge data sets on scalable machines, and the query optimizers [do] a terrible job of producing reliable, good plans [for these kinds of queries] without a lot of hand tuning. I think we need to totally rethink how we do query optimization, because the rest of [database] technology has improved, and the query optimization technology has not improved.

Do you have specific suggestions for how we should do query optimization instead?

I have an idea that sort of relates back to how Ingres did query processing, which was basically to iterate over the optimization and execution phases. Right now [the way database systems operate is that] we optimize, and then we execute. We [completely] optimize nine- and ten-way [join] plans based on ridiculous assumptions about [data] statistics. [The reality is that] after a couple of joins, you have no clue how many tuples are going to come out. You don't know whether [the values of the attributes in the columns being joined] are correlated or not correlated; you don't know whether your histograms are accurate---maybe you don't even *have* histograms. So [query optimizers] are making assumptions about joins five or six levels up in the tree based on just wishful thinking.

My personal view is that we need to revisit how we do optimization and execution. Right now, we optimize and then we execute. Instead, I think we need to look at something [like,] for example, optimize a little bit, execute a little bit, optimize a little bit more, execute a little bit more. [We should just try] something different, because that's one area where the technology has not improved.

And this is not to say that Pat [Selinger] didn't make a huge contribution in her work. When you write one paper and that finishes off the field, that obviously is a superstar paper, and Pat is a true superstar! But now we've gotten so capable at the execution side, we need to go back and redo optimization. And just adding better histograms is not going to solve the problem. I don't know how to do it, but that's one [direction I think is important].

[When you] singled out query optimization and I/O[, what did you mean by I/O?]
What I mean by I/O is that disks keep getting slower and slower. If you actually look at the transfer rate, disks are getting faster; but if you divide transfer rate by capacity, disks are actually getting [slower.]

Some people will advocate that what you ought to do is put a SQL processor in the disk controller, [creating] an intelligent disk. I think that's not going to help the problem; I think intelligent disks look just like an old database machine, where you had a processor and a disk together.

We're looking at [an approach to this problem] at Wisconsin: we're trying to see if we can make vertical partitioning work. It's a very old idea; the Bubba project at MCC did it, [calling it the] decomposition storage model. […] The idea is if you only need one or two or three or a handful of columns from a table, why do you read the whole [table]? Vertical partitioning makes hardware caching work really well; it makes compression easy to do; it may greatly increase the effectiveness of the I/O device that you're trying to use.

[O]bviously, as database people, we can't go and change how the disks get manufactured. We have to live with commodity disks. And they [are] going to be a half terabyte in a couple of years, a terabyte two years after that; by 2010 they'll probably be a couple of terabytes. Databases aren't growing as fast as disks are growing, unless you are doing imagery or video.

[In sum,] I just think I/O is a big problem, and right now vendors just throw disks at the problem, because disks are becoming so cheap. Maybe there's something interesting that we can do in the I/O area.

Any other core areas that you'd like to single out as needing extra attention?

I'm sure there are others, but those are the two that I been thinking about recently.

I think there's a fundamental problem in the way SIGMOD and VLDB papers get reviewed. I had a paper recently that was rejected from SIGMOD [but received a] best paper [award] at VLDB. … I think [the randomness of the process] must be very hard for junior faculty members.

Do you have any favorite hot areas? Bandwagons that you are glad to see people jumping onto?

Obviously XML is a hot area. The thing that I think is interesting about XML is that the [database] community failed to do distributed relational database systems, and I think XML is neat because if it really happens, and people serve up XML and their web sites run XQUERY, you can think about building a big gigantic distributed database system [on top of that]. I think that's a pretty exciting area that the community can work in. [I] think solving distributed database problems in a huge scale will be an interesting challenge for us to work on over the next few years. [But] there are already too many people working on XML and XML databases.

[XML databases] is not a core area, but I guess that is the hot area [that I would single out as particularly interesting]. And that leads into the question: can we do something that connects with the artificial intelligence community with regard to semantics? [J]ust XML alone is not going to do it, in terms of being able to do something intelligent with large amounts of data [that need to be integrated].

There's a feeling among some people in the database community that students are publishing more delta papers than they used to, because it's easier to get a paper published at a top conference if it's a delta paper, because it's easier to plug all the holes that a reviewer might complain about in a delta paper, and students have to have many more publications than they used to, if they want to get a good job. Is this really happening? Are there more delta papers now than there used to be, and if so, does it mean that there's a problem, and if there is a problem, what is the solution?

I'm not sure that there are that many more delta papers.

I think there's a fundamental problem in the way SIGMOD and VLDB papers get reviewed. I had a paper recently that was rejected from SIGMOD that [received a] best paper [award] at

VLDB. The paper was basically unchanged between the two submissions. Now, there's something wrong if [a] paper is rejected by one conference and [the same paper is] recognized as a good piece of work by another conference.

I don't know what's gone wrong with the refereeing process. I think it's becoming a random event [whether] you get a paper accepted or not accepted. I think we either need to introduce a cycle of feedback into the refereeing process for conferences, in which you would submit your paper, the program committee would review it, would give you the comments back and give you a chance to rebut it before the program committee met; or we need to go through a multi-round process.

[I] think [that] right now the process of getting a paper accepted is a total crap shoot. I think it must be very hard for junior faculty members. As a senior faculty member, I get frustrated when my papers get accepted, and [that's true even though whether they are accepted] doesn't matter [for my job prospects]! Especially since I'm the chair [of my department, so] the dean sets my raise (and I don't get reviewed by my colleagues), and the dean doesn't look to see whether I had my two VLDB papers rejected[.] But for a junior person who is untenured, it must be very very unnerving to see what you think are good papers rejected---and for reasons that are not clear.

The fix that you mentioned sounds a lot like the journal refereeing process. Are you proposing that SIGMOD turn into TODS, say?

Certainly not, because [TODS] is nothing but theory papers these days.

> I've been going to SIGMOD since 1979 [and] it's always the same---let's do something different for a change! … I think it would help [if] SIGMOD were twice a year

I hope not!

The journal process is open ended, but [the] program committee process is not open ended. Right now the time line is absolutely ridiculous. We submit papers the first of November and we publish in June. That, by my count, is eight months. We all know that the papers are already typeset. The process of going from camera ready [copy] to production is a non-event. There is a long window, from basically November 1st until March or April, over which we could carry out the refereeing process. It wouldn't be like a journal, because it's just one round [of review and discussion]. You submit your papers; you get your comments from [the] reviewers on the program committee; you have a chance to write a rebuttal to the referees; and you don't change your paper [during this process]. And then the committee acts.

[I'm suggesting this alternative because] I think sometimes committee members either review papers [in areas] that they don't know very well, or they misinterpret what someone writes. I think we need to try something a little bit different, because I think there's too much uncertainty in the process in terms of what papers should and should not get accepted.

I also think that we should accept more papers than we present. Some people give good talks; some people give bad talks. I think it wouldn't hurt us to[, for example, out of] 250 papers submitted to SIGMOD, maybe [accept] 75 or 100 for a proceedings ([rather than] 50), and then

pick 25 to 30 to [be presented in talks]. I think [that] you don't necessarily have to present every single paper that is [accepted]. Some papers would make better presentations than others.

I think that the worst thing a junior faculty member can do is spread himself or herself too thin across multiple areas. … I think a junior faculty member should not have more than three or four students at the very most

When you pick those 25 to 30 [papers], how would you know that you were picking the papers that had the good presenters?

I don't really know. I just think, let's do something different! It's like introductory computer science classes: we've been doing the same thing [for so long---] I've been going to SIGMOD since 1979 [and] it's always the same---let's do something different for a change!

Would it help if SIGMOD were twice as big, had twice as many acceptances; [would it] make [the acceptance process] less random?

I think it would help [if] SIGMOD were twice a year, or if VLDB were some place more reasonable than Hong Kong or wherever it's going to be next year, which is very far away.

Well, that [choice of locale] is very reasonable for everyone who lives in Hong Kong, of course.

[Yes, it's] very reasonable for everybody who lives in Hong Kong, but most of the people come from the United States and Europe. Organizing two SIGMOD conferences a year [would be] difficult [under the current system,] because [currently] we have [each SIGMOD conference] someplace different. Big organizations have trade shows, and they hire people to run their trade shows[, which are often in the same place each year]. Doing the program committee part of SIGMOD or VLDB is not that difficult. [The hard part is to do] all the local arrangements. I think that we have enough people in the field that we could stand to have an additional conference in the United States every year.

Sounds interesting.

Do you have any words of advice for fledgling or midcareer database researchers or practitioners?

I think [that my words of advice are] no different than I'd give to any junior faculty member. (Being a department chair, you have to worry about these things.) I think it's important to pick one or two areas and do a really good job in whatever areas you pick. I think that the worst thing a junior faculty member can do is spread himself or herself too thin across multiple areas. If you want to do data mining, well, become one of the best people in data mining. Don't try to do data mining, cubes, XML, [and] main memory databases. Pick one or two areas, and focus your efforts on those areas.

My other piece of advice is, don't take on too many students too early. I think a junior faculty member should not have more than three or four students at the very most, because students are a great resource and if you have too many of them, you simply can't work with them in an effective fashion.

I think CISE puts far too much money into supercomputers and terascale and grid computing and all the things that [the interviewer's school] gets.

How many [students] do you usually have?

Too many! I currently have seven or eight, and I'm trying to get back to three or four.

Seven or eight PhD students?

Mostly PhD students and a couple of undergraduate students. I'm starting to get more and more into hiring undergraduates.

They can be useful sometimes.

They can be very useful.

If you could do one thing at work that you're not doing now, what would it be?

[I] don't have a good answer for that question… go to the pool and swim more?

I never was smart enough to do database theory work.

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

I wish I had a stronger mathematical background. I think there are a lot of things I don't understand that I wish I could understand. I was a chemistry major as an undergraduate, so I didn't take a lot of math [courses]. I think there's a whole set of research that goes on that I simply can't participate in. That's one thing I guess I wish I could change.

Are you saying that if you had this background, you'd be doing more database theory work?

Possibly. I never was able to [do that kind of work]. I never was smart enough to do database theory work. I have one PODS paper, which sometimes people kid me about. But it was the student's [paper], it wasn't mine. […]

Thank you very much for being with us today.

Thank you for hosting me.

[TODS] is nothing but theory papers these days.

[We] failed to do distributed relational database systems, and [XML] is neat because if it really happens, … you can think about building a big gigantic distributed database system [on top].