Phil Bernstein Speaks Out
on Trends at Industrial Research Labs, What Metadata Management Products Need, Peculiarities of the Tenure System, How to Fix the Problems with Database Research Conferences, and More

by Marianne Winslett

Phil Bernstein
http://research.microsoft.com/~philbe

This interview with Phil Bernstein was videotaped at SIGMOD 2002 in Madison, Wisconsin, and revised slightly in July 2004 to bring it up to date.

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 are in Madison, Wisconsin, the site of the 2002 ACM SIGMOD and PODS conferences. I have here with me Phil Bernstein, who is an ACM Fellow, a recipient of the SIGMOD Innovations Award, and a member of the National Academy of Engineering. Phil is perhaps best known for his seminal work on transaction processing, especially in the area of distributed databases. Phil is on the Board of Directors of the Computing Research Association and of the VLDB Endowment. He did his PhD work at the University of Toronto in the 1970s. Phil was a professor at Harvard University and the Wang Institute, and a researcher at Digital Equipment Corporation, before joining Microsoft in 1994. At Microsoft, Phil has worked both in product development and in the research division. So, Phil, welcome!*

Thank you.

Phil, in the 1970s and 80s you and your colleagues figured out how to do distributed databases. And then distributed databases just seemed to vanish from the picture! Does that represent some kind of failure of the research community? Does it mean that the work was a waste of time? Or do you see that work as being beneficial to us today?

I think it's quite beneficial. I think at the time we imagined there was a product category for distributed databases. Instead, it has just turned into being a standard feature for any database system. In a cluster environment today, you expect to be able to run multiple databases and to view them as a single database. You expect to be able to run transactions that update data on multiple nodes of the database, and to ask queries that span multiple nodes. With respect to heterogeneity, you expect to be able to link up to databases at other locations and treat them as all part of one big database that you can query. Another dimension of distributed databases was replication, which got quite a lot of attention from researchers. Replication subsystems for today's major database products are almost systems in their own right, they're so highly functional and complicated. So I think the work on distributed databases was actually quite a big success.

So it all happened, just not in quite the way that people had thought it would.

It rarely unfolds the way that you expect.

[Customers are still saying,] "Do we really want to commit to this [new metadata product], or wait until some other guinea pig has survived the ordeal?"

In a recent interview with the ACM Ubiquity *online magazine, you spoke about metadata management, one of your current areas of interest. I think that you referred to metadata management as the Rodney Dangerfield of research topics--- it just don't get much respect--- and said that metadata products had not met with long-term success yet in the marketplace. Is the problem that metadata management needs a killer application, or are there research issues that need further attention, or is the problem somewhere else?*

Time will tell. Eventually metadata products will become more successful, because the problems continue to be with us and they will someday get solved. As a guess, I think what has been missing is a system facility that enormously simplifies the users' and the application developers' approach to metadata management. We have made a number of runs at the problem: data translation in the early 70s, database design theory and entity-relationship modeling in the late 70s, object-oriented databases in the 80s and early 90s, and ETL [extract, transform, and load] tools for creating data warehouse loading scripts in the late 90s.

In the case of object-oriented databases, the metadata was structural information and often engineering design information, and its treatment certainly represented a step forward. There are a lot of good things about object-oriented technology, but it hasn't really taken over as a major approach. Instead some of the object-oriented technology is being rolled into relational databases. As a product category, object-oriented databases haven't taken a big enough step forward for users to feel that they just *have to* use it.

I think that this is actually a pervasive problem: if you want to introduce a new technology, you either have to roll it into some existing product category that already has a lot of customer acceptance, and make it available over a period of releases; or else you really try to create a new kind of product, as the relational database system vendors did in the early 1980s. Of the two possibilities, creating a new product category is much harder to do because you're either trying to get somebody to agree to a new kind of product that they've got very little confidence in, or else---perhaps even worse---you're trying to displace some other product that's currently filling that need.

With respect to metadata products, no single product has enormous market share, and no product has a really compelling value proposition. Every time a new product comes out, people look at it a little bit skeptically, saying, "Oh, here we go again. This is the fourth time we've seen a new generation of purported solutions for metadata management. Do we really want to commit to this one, or wait until some other guinea pig has survived the ordeal?"

So are you thinking that perhaps metadata management functionality will be merged into existing products?

Maybe, but I do think that we need a much bigger step forward in functionality too. If we think back to the initial relational database system products, they were not good systems: they were buggy, they were unreliable, they were slow. But users only had to write 5% of the application code that they would have had to write if they had used the competitors' products. So some users just had to swallow those unpleasant aspects of relational products because otherwise they couldn't write the application---they couldn't afford to write twenty times as much code. If we had a solution for metadata that had that kind of characteristic, that made it just so much easier to solve your metadata problem, then you would be willing to live with all of the flaws in the initial metadata management product. When we reach that point I think we will have a market opportunity, and that's essentially what I'm personally trying to do---to take a big step like that. But we'll see, time will tell.

The initial [RDBMS products] were not good systems: they were buggy, they were unreliable, they were slow. But users only had to write 5% of the application code that they would have had to [before].

You are one of a handful of people who have spent considerable time in academia, in an industrial research lab, and in a product development group. Product development groups aren't going to go away any time soon, academia may undergo some major transformations but will probably continue as a hotbed of computer science research, and industrial research labs are somewhere in between. What is the future of industrial research labs? Are they an outdated concept, as some have said?

I don't think that they're an outdated concept. They already have undergone quite a lot of change in the last ten years. First of all, an industrial lab has got to be in a healthy financial environment. It has to be in a company that has plenty of money to spend on research and can continue that cash flow for a long period of time, because research is not the kind of activity that can suffer major swings in budget without falling apart. If you look at the big companies that are currently making a lot of money, they have very healthy research labs---IBM, Microsoft. In other companies that are having more financial problems, their formerly very healthy labs have undergone changes in response to the difficulties of the parent organization. These changes are not due to anything bad going on in the lab. The lab is as good as it always was, but the parent organization is no longer able to afford the same level of activity.

The change that all the labs have experienced in the last ten years is that the time frame for success of industrial research has been shortened. There is less really exploratory university-style work going on in industrial research, and arguably that is okay. We do have a large capacity to do very exploratory research in universities, and maybe that's where that work ought to be done. Clearly there is some highly risky research that goes on in industrial labs as well. It is just a relatively small fraction of the overall lab budget. There was an article written some years ago by an AT&T Labs researcher, Andrew Odlyzko, titled "The Decline of Unfettered Research," claiming that even universities have been affected this way, so it's not uniquely an industrial lab problem either.

So is the shrunken time frame a problem, or just a natural transformation and not a cause for concern?

I don't think it's a problem that industrial labs have this shorter time frame. After all, in essence, you're spending the stockholders' money and this is not a philanthropic enterprise. In a university, to the extent that this trend holds, maybe it is a problem. If university research projects are not pushing the envelope a little further, trying harder to take a bigger step than an industrial lab would take, if that's really true overall---I'm not sure that it is true, but if it is then I would regard that as a problem. In the database area, I can point to a fair number of very exploratory kinds of activities going on in universities, so I don't think we're in a particularly unhealthy situation right now.

The [tenure process at Harvard], and I think in many other Ivy League schools too, was heavily biased toward avoiding false positives.

I remember talking many years ago with a very distinguished member of the database research community who said that all the most important ideas of the previous five to ten years could be traced back to you. Yet many years ago Harvard chose not to offer you tenure. Does that mean that the tenure system is irreparably flawed?

Oh, I don't know. This is a perennial problem. My particular situation in Harvard was definitely a unique thing about Harvard at the time. The process there, and I think in many other Ivy League schools too, was heavily biased toward avoiding false positives. If there was *any* doubt, if anybody expressed a plausible argument for why the university might not want to tenure this person, then the university's response was to deny tenure and wait for another candidate to come along. The attitude was "we don't really need to take this risk." That's apparently what happened to me. This attitude has loosened up some in recent years, at least at Harvard.

I don't think the tenure system is irreparably flawed, but it does have its unfortunate aspects in that it steers junior faculty in directions that may not be not the ones they would take if they didn't face this important hurdle in six years. The tenure system puts universities in the odd position of having to make what they recognize to be really critical decisions based on limited information. They are hiring somebody for life, and maybe they really don't want to be doing that, but that is the system. So you can understand why some departments would tend to be overly conservative.

Does that mean that maybe more departments should adopt Oregon Graduate Institute's model and hire people on contract? Do you it could be better in the long run?

To be honest, I'm not sure it would make a whole lot of difference in computer science, because people are so mobile and the current job market is so good. I don't know whether people will feel that way in twenty years time, if the job market turns around. There are advantages to having people on contract, to putting a little bit of more serious review in the cycle. People should be expected to do good work throughout their career.

The tenure system was originally created so that people with politically unpopular ideas would not be subject to indiscriminate firing just because their colleagues didn't agree with them. This idea still has merit, so hiring people on contract certainly has its downside.

Is research into core database server technology all played out, or are there interesting research areas that people should still be looking into?

Query processing is an intractable problem, so it will never go away. It's a really good area to get into, because you know there is no solution---you can work on it forever! The servers are extremely complex. There certainly is room for people to rethink how to put the system together to make it simpler, and that work is going on. A lot of this kind of work goes on within product development groups, just because they have a hard time coping with the engine complexity that they're faced with.

You submit a paper to one conference, and it gets rejected. You submit it to the next conference, and it gets selected as one of the half dozen best papers, with hardly any change other than the fact that you got three different referees this time who were more appreciative of your work.

Every now and then, a new paradigm comes along that will create some new aspect to the engine. Right now, one such paradigm is continuous queries and streaming data, which are likely to require incremental enhancement to engines. Another is the ability to handle very large numbers of triggers, which is kind of a different spin on the same problem. But certainly it's not like it was in 1980, where we didn't know how to do anything very well and all the problems were open to us. It is not like that any more.

How is the situation right now for database conferences? Do we have the right number of venues? Do we have the right criteria for acceptance? Are referees acting appropriately? If there are problems in the system how should we go about fixing them?

There is definitely a problem. First of all, we don't have enough bandwidth to publish in a timely fashion all the good papers that will eventually get published. We're only accepting 1 out of 6 or 7 papers in SIGMOD and VLDB, and the submission rate is going up. So we need to change something. The second (and related) problem is that the reviews are not consistent; they have very high variance. You submit a paper to one conference, and it gets rejected. You submit it to the next conference, and it gets selected as one of the half dozen best papers, with hardly any change other than the fact that you got three different referees this time who were more appreciative of your work.

There are lots of proposals on how to fix the second problem. Maybe we should be a little less selective: acknowledge that people do have different tastes, and just accept more papers. It's been suggested that we greatly increase the number of papers we accept per conference, and have poster sessions and reduce the number of formal presentations in lecture halls. I think that's a good idea and it’s starting to happen.

Another thing I would like to see is a online journal with a review turn-around schedule similar to that of conferences, but with a feedback loop where referees actually have to defend their points of view on the second round and an author can systematically argue the points one by one, with an editor there mediating. And maybe in order to keep the turn-around time short, this journal should also keep the papers of modest length. One problem with high variance reviewing is that you never really get a chance to respond to the reviews. You edit the paper according to the criticisms of the people who reviewed it the last time you submitted it to a conference, but of course they're not the reviewers the next time, so you get a new set of criticisms to deal with. The result is that the process never converges and good work doesn't get accepted. A variation on this idea is to allow borderline rejections to conferences to be revised and resubmitted to the same reviewers. If the revision is accepted, it appears in the next conference---maybe with some linkages between conferences so, for example, a rejected SIGMOD paper that's quickly revised and accepted by the same reviewers could appear in the next VLDB. Since ultimately the whole point is to get good work published, I think that we need to start being more a little more creative and try some different things.

[This topic has gotten hotter since this interview two years ago, as evidenced by the panel session at SIGMOD 2004.]

Have you thought about having VLDB or SIGMOD twice a year instead of once a year?

No, but I don't think that would work. I think there are enough conferences. It's a burden for people to go. Travel is expensive, and it can be disruptive to their lives. A lot of the reason people go to the top conferences is not so much to view the paper presentations as to have conversations with top people. There is a limit to what you can do by email and by reading each other's work, and you want to get your ear to the ground and hear what everybody is up to. So if the whole goal of having more conferences is simply to have more room for publishing, let's just abandon paper proceedings, hand out CDs, double the number of papers we accept, and be done with it, because that will solve the problem less expensively.

As program chair for VLDB 2002, can you comment on VLDB's strategy to broaden the range of papers included in the conference?

At the recommendation of the VLDB Board of Trustees, we introduced a new split between the core database topics and infrastructure for information systems topics. The Board's feeling was that we were not accepting enough of the middleware-oriented, application-oriented, enterprise-computing-oriented papers and that stimulating the program committee through persuasion to accept more of these papers was not really going to change things. The only way to change things was to split the committees so that those papers were not competing against well-formed papers on a new query optimization technique with nice performance measurements and beautifully put-together graphs and equations. There are a lot of other conferences on database-related topics that are not core database conferences but that database people have migrated to just because their papers were not getting into SIGMOD and VLDB. Hopefully this will start bringing some of those people back into the fold and we'll be getting more of those papers in the core database conferences instead.

So is this the first year that VLDB has tried that?

Yes, [2002] is the first year.

And how do you think it's worked out this first year?

I think it worked out reasonably well. We certainly got a lot more information system infrastructure submissions than we've gotten in the past. We also have a larger total number of submissions that than we've ever gotten before, which is especially surprising given that the conference is in Hong Kong, which you might imagine to be an impediment. People might feel like it was an expensive trip that they were not likely to take. I guess we'll see what the community thinks of the pool of papers that were actually accepted and whether that works out.

What are the advantages of having separate parent organizations for SIGMOD and VLDB (ACM and the VLDB Endowment, respectively)? Why not just one big organization?

Part of it is historical accident. VLDB started as an independent activity and eventually turned into a foundation as a way of perpetuating the conference. I think there is a role for an internationally-oriented database conference every year. It is a different group; there is some overlap but there is a different feel to VLDB. You get a lot more international participation. It's held in nice places all over the world. It---

But you are talking about the VLDB conference. What about the organization?

You need an organization to run the conference, and it needs to have an international regulating body, and the ACM is largely North-American based. It would be hard for a North-American-based organization to have a conference where decisions were being made solely in order to enhance international participation. Would they really be serving their constituency by doing that?

[Editor's comment: In recent years, SIGMOD has been working to move to an international base. This effort includes SIGMOD/PODS in Paris in 2004 and abroad again in 2007 (most likely in Asia), a textbook donation program and the SIGMOD/VLDB Digital Library Donation Program for developing countries, a new program under development involving support for short courses and lectures in developing countries, enhanced international recruiting of SIGMOD volunteers, and other activities.]

What about the reverse direction, if SIGMOD seceded from ACM and joined VLDB? Would that work?

I don't know what is to be gained by it. The ACM has a lot of infrastructure to offer for SIGMOD. I haven't been involved in running SIGMOD, so I don't know what problems seceding might solve. But I do think it's a good idea for VLDB to remain independent of any of the parent professional societies. It's not constrained by a parent society, so it can adapt quickly and perhaps take more risky decisions. The VLDB Endowment is currently well-funded and does a lot of good work for promotion of database research internationally.

Can you comment on some of those database research promotion activities?

In developing countries the VLDB Endowment funds workshops and tutorials and provides libraries with VLDB publications. It created the VLDB Journal and has tried to be an innovator in accepting a broader range of papers than might normally be considered for an ACM journal; the VLDB Journal is now a highly-respected publication. These are all good things, and we're always looking for new ideas to spend time and money well.

I think a lot of the advice people give [young researchers is] a thinly veiled description of what worked for them, given their strengths and their organization. That kind of advice is not readily transferable…

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

I've noticed that there's been a lot of advice given to young database researchers at this conference [SIGMOD/PODS 2002]---

Yes, all conflicting!

Indeed, all conflicting. There are three pieces of advice I always give to people early in their careers. First of all, figure out what you're good at, whether it's teaching, theory, development, supervision. You're not going to be good at everything.

But you've done all these things.

I'm not equally good at all of those things either. You've got to figure out what you're best at, most productive at, and what you like the best. Conversely, you have to know what you do less well and need to either avoid it or practice at getting better. Secondly, you've got to figure out what activities are most synergistic with your institution, with your job, based on who your colleagues are, what kind of resources are available, what kind of students are available, and so on. Then you need to figure out a research program that fits those two characteristics, that exploits your personal strengths and also what your organization is like, and circumvents your weaknesses and those of your organization. I think a lot of the advice people give is very often a thinly veiled description of what worked for them, given their strengths and their organization. That kind of advice is not readily transferable, unless you happen to be a similar kind of person in a similar situation.

And third, you should be open-minded about exploiting unexpected opportunities when they arise: a new technology, a brainstorm, or a job offer. It's partly a matter of bravery, to devote less attention to your current successful path and do something new where you might fail. It's partly a matter of good taste, in deciding which opportunities are worth the risk. Taking a chance on pursing an unexpected opportunity is often the big break that leads to the biggest successes.

You've got to figure out what you're best at, most productive at, and what you like the best… I'm best at working in areas that are very messy…

So you tried all these different types of work. What were you best at?

I'm best at working in areas that are very messy, where the early abstractions have not been put together, where people are very confused and where a big step is possible simply by sorting through the mess almost as a philosopher would: taking a collection of loosely-related facts and questions, coming up with a model for understanding and lending some order to the mess, and figuring out a way to make orderly progress. That's the stage of a research problem that I'm best at. So in recent years I worked on things like TP monitors and middleware and now on metadata, which all have that characteristic. My choices were really very premeditated, based on learning over the years what kind of work I do best and like best.

Do you have a favorite piece of work that you've done?

Serializability theory. After working for awhile on mechanisms for distributed transactions, it finally dawned on Nat Goodman and me back in the late 70s that we really could account for the correctness of virtually all the concurrency control methods using a simple mathematical structure, and prove them correct and really reason through them quickly. Before we developed serializability theory, people would spend months trying to prove a concurrency control method correct. Now we could just look at the algorithm and in half an hour write down the properties of the execution histories it generated and apply serializability theory to figure out whether the algorithm worked or not. That was very satisfying, a lot of fun. There is not so much work in this area [proposing new concurrency control methods] now, but it's still fun to read papers and quickly do the analysis, to apply it yet again.

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

Experiment management.

You mean like scientific experiments?

Before we developed serializability theory, people would spend months trying to prove a concurrency control method correct.

Yes. Even in the software field, if you do any measurements, if you run any experiments, typically you run a *lot* of experiments. Nine times out of ten, the output of the experiments will be stored as files and viewed in a spreadsheet. After a week or two, you no longer remember what those old files are, what configuration you were measuring, etc., and you are not able to do any longitudinal analysis.

Every scientist who does experiments faces this problem. Some scientists have built fairly interesting systems to help manage their experiments. The database field has never treated this as a first-class problem, mostly because scientists don't have a lot of money to spend on database software. I think if you really looked around at the size of the market, you would find that there probably are a million customers for a system like that, counting all the people doing scientific measurement of various kinds. I think we'd benefit a lot from investigating this area. If I had time, that would be the next thing I would do.

Not coincidentally, this problem has a large metadata component---capturing formal descriptions of experimental configurations that evolve over time. This is a lot like metadata problems in the commercial sector, which involve capturing formal descriptions of database schemas, application interfaces, and mappings between them.

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

I'd be smarter. Other than that, in terms of skills, I wish I had become a strong systems programmer very early in my career. The popular myth back in the 70s was that doing development was a good way *not* to get a PhD, that you were going to do a lot of coding and after you were all done it wasn't really clear what you would have accomplished and why it was research. I took that advice and ended up not really developing my skill as a programmer back then. I feel that weakness has really hurt me over the years, in the sense that it took me a lot longer to really get the kind of systems intuition that I wanted. I wanted to be able to recognize good and bad solutions to systems problems, to recognize what really are problems at all, and also to be able to jump in personally and spend a couple weeks prototyping something quickly. Now I often have to rely on others to help me with that, because I'm just too slow at it.

Thank you for talking with me today.

You're welcome. It was fun.