Pat Selinger Speaks Out

on Why System R Was So Successful, Interactions with the INGRES and QBE Teams, How to Do Techology Transfer, How to Manage Difficult People, How to Mentor 3,000 People at Once, and More

by Marianne Winslett

Pat Selinger

Welcome to this installment of ACM SIGMOD Record's series of interviews with influential members of the database community. I'm Marianne Winslett and today we're in San Diego, site of the 2003 SIGMOD and PODS conference. I have here with me Pat Selinger, who is currently a vice-president of Data Management Architecture and Technology at IBM. Pat is the primary author of one of the most influential papers ever written in the database field: her 1979 SIGMOD paper, "Access Path Selection in a Relation Database Management System," set forth the guiding principles that underlie the query optimizers in every major database system today. Pat has also been instrumental in making sure that the research innovations at IBM find their way into IBM's product offerings. Pat is an IBM Fellow, an ACM SIGMOD Innovations Award recipient, and a member of the National Academy of Engineering. Her PhD is from Harvard University. So, Pat, welcome!

Thank you very much.

Pat, System R and INGRES showed the nay-sayers that it was possible to build a rational database management system with reasonable performance. As part of the early System R team, you were there at the birth of an industry. What were the biggest surprises that you had while building System R? Are there things that went spectacularly right or spectacularly wrong?

I came to IBM when the System R platform was just getting launched. There were people working on the [lower levels] of the system, and work on the [upper levels] of the system just had gotten started. I came with a background in operating systems and programming languages, but I knew nothing about databases. So they gave me Chris Date's book and said, "Here, read this, and then start interacting with the rest of us after you're finished." What was surprising to me was how much my background in programming languages and operating systems really matched up with what was going on in database systems, and how the database field went beyond the technologies of compilation and device-level drivers and management of things in memory and I/O. The database field capitalized on all of those other fields and then brought them together with a new spin. So for me personally, that was very surprising---to realize that I actually had something to contribute early on.

And for the project as a whole, were there things that went spectacularly right or spectacularly wrong?

I think we had a terrific group of people. At that time there were maybe 30-40 people who came in and out of the project, who were here on post-doctoral fellowships, or who joined us mid-stream in the project. The synergy, the feeling among the team of almost being family was really very enriching compared to the academic environment, where you're all by yourself working on your PhD. That was a refreshing change for me, and one that's really kept me at IBM because it's the kind of way I like for us to work together.

I didn't really have enough experience … to realize that System R was going to be something that people [within IBM] were going to take almost as is, and make a productized copy of it and [sell] it.

In hindsight, what do you view as the key elements in System R's ultimate success?

I think the fact that multiple companies picked up the SQL language and started implementing it was key to making SQL the pervasive database interface. And so that was an important ingredient in SQL's success. The second thing that I think mattered a lot is that we early on signed up a number of companies to use the system, and so as we were building they were using it. Boeing was one of the early users and they said, "[System R] is a tremendous tool, but our problem is that it is [only] February and we have [already] used up our computing [time] budget for the year[, from running so many database queries]!" And I said, "Oh, we're so sorry, we're really sorry!" And they said, "No, [don't apologise, System R] is wonderful. We can ask questions we could never find the answers to before. We're now able to do more than we could ever do before." So I think the boost in application development productivity for users due to System R was just a huge winner. And that the fact that we had customers telling IBM that this was important to them made a big difference [in the support we got from within IBM].

One thing that will be fun for our readers is that I asked Bruce Lindsay the same question that I just asked you, so when his interview gets published, they're going to get a different *interpretation of why System R was super successful and so influential. But I won't give away any secrets! I won't tell our readers what Bruce said! Our readers will just have to wait and see Bruce's answer when his interview appears.*

At the time that System R was built, it was impossible to realize what an impact it would have on the world, for better and for worse. With perfect hindsight, what do you wish had been done differently in System R?

In hindsight, I didn't really have enough experience as a system designer and developer to realize that System R was going to be something that people [within IBM] were going to take almost as is, and make a productized copy of it and [sell] it. So designing control blocks, designing interfaces and APIs to programming languages---we just sort of put together something that we thought would work. The error control block, the way that you pass variables and parameters into the system---those were not designed with thought and care and elegance. They were [just]

something that worked, and we didn't put them together in such a way as to make them extensible, to grow over time. We didn't think about a thousand other data types that you might want to express or represent. So I think in hindsight, I would have liked to do a much better job, a much more *thoughtful* job of that, so that System R's application programming interface would fit more naturally with the programming languages that used it.

Today the positions of INGRES and System R in the history of the database field are secure. At the time the systems were being built, though, the two teams were trying to solve the same problems with the same goals at the same time, making rivalry inevitable. What forms did the rivalry take?

I think it was a healthy competition in terms of exploring different aspects of the [design] space: different ways of representing views, different ways of doing query optimization, and so on, to the benefit of the [database] community. We got to explore a much richer set [of approaches] than any one of the projects would have done on their own. So I think it's healthy for the industry that we were able to do that at the time; I think it moved us along faster.

With hindsight, if you could go back to those days, how do you wish that the situation would have been handled? I ask this because today and in the future, other researchers will find themselves in somewhat similar situations and may find your comments helpful. (For our readers, there's a more detailed description of what happened back then available on the web in a marvelous document produced by the 20-year SQL Reunion, with a lot of fascinating history about System R and even also QBE and the interactions between the System R team and the INGRES team.)

I think there are areas where we could have done a better job of of having discussions at venues like SIGMOD---going through some of the pros and cons of the technologies. Had we had a more open discussion, shared among many people within the community, I think that would have been [better] than [the two groups] kind of staying on their own and not interacting as much.

Inside IBM, QBE and System R were often viewed as rivals. With perfect hindsight, is there a way that that situation could have been handled better?

I don't know that there was a flaw in the handling of it. There were two groups working on different aspects of a similar problem, and some people thought there was a competition there. I think in hindsight that the QBE work really focused on the user interfaces and how people formulate queries; they made it so easy to do easy things. SQL and the System R work was really more oriented toward the "heavy duty" stuff and that's where where we focused our efforts: on the deep technology, the scalability, the performance, and so on. That wasn't really a focus ever of the QBE project. I think had someone rationalized that in a better way earlier on there might not have been been as much perception of conflict or feelings of conflict.

So somehow people didn't realize that those *were the two contributions, one at the user level and the other---*

There never was was a bake off or a head-to-head---

What about the Shootout at the OK Corral?

What about the Shoot Out at the OK Corral? (Readers who are interested in the Shoot Out at the OK Corral can read about it in the 20-year System R Reunion documents on the web.)

You're right. I'm getting too old! That was a performance race.

Which doesn't seem like the right---

Given the hindsight, given who focused on what, performance was not even the right thing to even *look* at when comparing the two systems. What we should have been trying to do is to glue the QBE front end onto the System R engine back end.

Yes, and of course System R would have beat QBE in a performance race. So I guess the gluing never happened, although---

Well, no, that gluing-together is exactly what we did with the QMF product, because the interface of QBE and QMF on top of DB2 was really the product set that came out the door in QMF. So the gluing was done, but it was done in the product lab and not in the research lab.

Oh, I see.

IBM has been uniquely successful at transferring research results into products. For many years you oversaw this effort. What are the secrets of successful technology transfer?

I did two [special] things when I set up [the Database Technology Institute, IBM Research's organization dedicated to database technology transfer]. I went to every manager in the IBM development community in the database area, and I sat down with them one-on-one for at least an hour. I talked to them about what were their goals, what were their problems, and explained what I thought this group was going to do and how I was going to interact with their team. This eliminated up front the fear factor of "I'm in competition with you." The second thing that we did was that we developed the ideas together jointly in a meeting where we talked about what should we work on, what are the problems that are important to the development people, what should be the agenda for the coming year. And we had people on the research team who went down and lived in the development labs and spent their days worrying and helping and participating and generating ideas [side by side with the development people], so that the researchers proved their "worthiness" in essence. By the time [all these] things were done, there was a mutual respect. So the way to develop that kind of team work is to just rub elbows.

The yearly meeting: was that the product people telling the research people what their problems were?

Well, it was a week long meeting. And the first part of the week was the developers telling us, "here are my development plans, here are the things I think have under control, here are the things that I think need a longer period of time or more invention than we can get done in the scope of a normal product release." And then we got back to the Database Technology Institute and talked about "here's what I did last year, here are the things that I contributed," and then we sat down at the end of the week and we made a ranked list of which things we wanted to invest effort in during the coming year, which are more important to us and to IBM. Things that multiple product groups found valuable went higher in the list.

So, among the needs of the product groups, was it a match-making exercise between what you had and what they needed?

Yes.

So it wasn't that you saw their needs and then you went off and did the research to meet the needs?

We had the entire Database Technology Institute team in the room all week listening to the presentations, and people interacted all week during breaks and dinners, and so forth, getting to know each other. We would have discussions like, "Well, how about this approach to your problem?", "I bring this skill to bear on your problems", and "How about if we solve the problem that way?" And people had a better idea at the end of the week about what whether they had something relevant to offer.

In other companies that you read about where [technology transfer doesn't work] very well, what tends to happen is that researchers stop putting in effort once the picture's on the blackboard or the paper is written or the system is built.

And the people in the Database Technology Institute, were they full-time employees there or were they full-time researchers in the research division who also belonged to the Institute?

The Database Technology Institute was a club. There were no people working for me. This was all done based on people wanting to participate. So it was a volunteer effort and the way that we kept things interesting is that we found interesting problems for people to work on, and there was a willing enthusiastic consumer for the results of the work. Most projects had a mix of people from development as well as people from the research team.

And one more thing: you mentioned dispelling the competitive aspect. Could you explain more about the competitive feelings that could arise?

In other companies that you read about where research doesn't work with development very well, what tends to happen is that researchers stop putting in effort once the picture's on the blackboard or the paper is written or the system is built. Their attitude at that point is, "Here, I did this, don't you love it?" And that automatically generates a response of, "Well, I could have built that better, I would design it differently, you don't understand my world, you don't understand my customers." If it's an after-the-fact sell, it's always harder. There's a big mountain of emotional engagement that has to happen that's missing. So, if I get you involved from the very beginning, and you feel like you're helping shape this and direct it, and telling me the things that are important, then it's naturally going to be a better fit.

What led you to join industry and to stay at IBM, as opposed to spending some time in an academic position or working for several companies?

I interviewed in academia as well as industry. I was determined to work in California because I lived in Boston for eight years and it was cold and slushy and snowy and it was so much nicer in California. So I was convinced I wanted to find a job in California. I had job offers from UC San Diego and so on, and I had a job offer from IBM in San Jose. And I looked at the people who I would work with and the kind of work I would be doing. In my RA position at school, I had done some team building where we built a language compiler, and I loved doing that. I loved being part of a team. And at least at the universities at that period of time, it was an every-professor-for-themself kind of mechanism. Yes, as a professor you have your graduate students, and they help

you, and you become a team with your graduate students, but it's not the same as having fifteen people who always stay there to work with. And that was just very appealing to me and it remains so to this day.

You haven't switched between companies like some System R people did. So what led you to stay at IBM?

I like the people. Top-notch people. So that's the principal driving reason. And I like having the impact that we've had in the industry, and I see the potential for more.

I think the hard transition for a fresh PhD to make is from the mind-set of individual research, where I have to prove that I did this by myself …, to one of being part of a team in an industrial research setting.

I understand that you've been very involved with mentoring people at IBM. What can one do to help turn a fresh PhD into a researcher whose work has significant impact?

I think the hard transition for a fresh PhD to make is from the mind-set of individual research, where I have to prove that I did this by myself and I have to invent [everything myself], to one of being part of a team in an industrial research setting. That is a difficult transition for some people because they're not really trained to work on a team. They don't understand necessarily the discipline of I-add-code-only-when-it-doesn't-break-your-code software development techniques. Within the research divisions, it's still five-to-ten-people kinds of projects. But if you start working with a development team as part of, say, the Database Technology Institute, all of a sudden you might be working with 200 to 600 people and contributing software to a global entity that's just way bigger than than what you can get your brain around, certainly at first.

So how do you help them through that transition?

What we have done is we have this concept that Hamid Pirahesh invented, called immigration projects. And these are projects that are pretty well contained, that you only have to know a certain part of the system to do. You also have a mentor who helps you through the process and helps you build your immigration project. So, that way you get used to doing this kind of software development, this kind of design, in a way that's a little bit more structured than a free-wheeling researcher/developer might do as they become more experienced.

You have been Jim Gray's manager and also Bruce Lindsay's. Jim suggested that I ask you about the secrets of managing difficult people such as Bruce and himself. How do you do it?

Well, I not only managed those two but also Franco Putzolu, who is an Oracle fellow. All three of them worked for me at the same time. They were my only three employees and they were my first employees as a manager. So they broke me in and they were, yes, enthusiastic, passionate, full of ideas and not necessarily agreeing with one another. I think the secret there is that I just really liked them. I really respected them. You know, the more you take care of people, the more the people just kind of get the work done.

Well, maybe so, but one of those three people told me that you can get anyone to do anything you want them to do. So what's the secret to being able to do that?

Well, it's---I just like them.

You know, the more you take care of people, the more the people just kind of get the work done.

[Bewildered.] Liking them is...

Well, it's that I care about them, I try to be persuasive, I'm a consensus builder, not a dictator in my management style. And so that's really how this Database Technology Institute succeeded. I pull people together and get them working on the same thing in at least more or less a common direction.

But now that you're a vice-president, you must have hundreds of people under you and you can't like every single one of them, can you? So how do you do it now?

What I do now is that I essentially become the team mom for the 3000 people in database development in IBM, and they know they can call me. They have my home phone number, they have my cell phone number, and they can come to me and we can have a private, off-the-record talk about whatever's bugging them and I will help them. So, really it's the care and feeding of people and caring about them, their careers, their having interesting work to do, and getting them to see the bigger vision and understand that they're a part of that. That's really what I spend a huge amount of my time doing: pulling people together to get them to have a shared vision. If you build a shared vision then everything else kind of follows from that. So, none of [these 3000 people] work for me---well, maybe 20 of them work for me---and I can't do [this shared-vision building] with the entire [set of them, including the] junior programmer who just joined last week. I end up doing this [vision-building] with the senior technical leaders---the top two tiers of people, in terms of management and leadership and architects---and they then fan out and do it with their teams.

I essentially become the team mom for the 3000 people in database development in IBM, and they know they can call me. They have my home phone number, they have my cell phone number…

So if 3000 people have your home phone number, how often do you get calls to that number---just so I can calibrate?

It's a one-or-two-emails-a-week kind of thing, that level of involvement. When I'm not away on a trip, I spend nearly every lunch that in a mentoring session with one person or another. I have maybe 30 or so people that I mentor intensely. And I also have a larger group of casual-discussion-whenever-you-have-a-problem kind of folks.

And the people you mentor are drawn from certain ranks or certain sections?

I don't pick them; they pick me.

Oh, they pick you. Okay.

Some of them are very junior, you know, two years with the company. But I think they have really good talent. And others are people that have been there nearly as long as I have.

What led you to want to be a manager rather than a pure researcher, and how is the transition? Maybe you have tips for other people who make this transition.

I made that decision early in my career and in hindsight, that probably was too early. I was only with IBM for three years and I'd done exactly one project, System R, when I became a manager and underwent trial by fire, as you pointed out before, with Bruce and Jim and Franco. But the reason I did it is that I looked at what I could get done with a group of people, which is so much more than I could get done just with my own two hands. So, that was my motivation. I spent three years as a first line manager, doing the R* distributed database project, and then catapulted up to being a fourth line manager and ran the computer science department for three years. I think that rise was way too fast, although I enjoyed the job when I had it. It was like being at university and taking 22 classes, because there were 22 different projects that were going on in the computer science department, across a real variety of areas from theory down to disk drives and printers and databases. It was interesting but it wasn't as fulfilling [as I would have liked]; you couldn't get deep into any particular technology and I found out that I missed that. I created the Database Technology Institute, while managing the computer science department. The more time I spent structuring the Institute, the more I thought, "I want that job!" And so I went to my boss and said, "See that job over there? I want that job, I want to move out of this one and go do that, because I have this vision of how I want this to happen and I know I can make it work." And so I've combined this management capability with the technology depth that I really love.

And do you have any tips for people who are currently moving from engineering or research to being a manager?

What I tell people that I mentor today is do not make that move until you've got some achievements under your belt technically. Make sure you understand what a successful project is and what successful technical work is; make sure you understand that, because that's going to be your reference point for the rest of your life. So, wait a little while longer, don't rush into management. Get yourself a big stack of things you know you have done and that give you some variety of experience. After that, there'll be plenty of time. If you're good at managing and leading, people will ask you every six months to take a management job. This won't be your last opportunity ever in your life.

> [Before you become a manager,] make sure you understand what a successful project is and what successful technical work is, … because that's going to be your reference point for the rest of your life.

There's a feeling in the database research community that IBM's database researchers have become too product-oriented. Are you concerned about such an impression being formed? And are you in agreement with that assessment? If it's true, how will IBM Research make sure it doesn't miss out on inventing the future of database paradigm shifts?

What happened---this was deliberate on our part---in the mid-90's was that we were very, very late to the party in terms of databases on UNIX and Windows. And I went to Janet Perna in Toronto, who at the time had the responsibility to build the Windows and Unix database product. I went to her and I said, "Janet, I've got a research team here who is just not going to be relevant to anybody if we don't have a product entry in the UNIX and Windows space. So we are willing to take our Starburst technology and put our research pencils down and help you build this database product and get it out the door." So we deliberately put the researchers to work doing product development, and I think we've fundamentally changed the personality of the research team in doing that. There are some people who still today love being able to check in code to the current release. So we have that extreme. On the other hand, we are trying to pull some people back from the development side, to do the things that only researchers can do. And that's a very

deliberate decision as well, to foster the new only-researchers-can-work-on-this-stuff areas. But the reason that we're perceived as being very product-oriented is that that's the heritage, that's what we had to do to stay competitive and to make it worthwhile for IBM to have a research team in databases.

Pat, your PhD work was not in the database area. At this point, is it possible for a person without a database background to move into database research?

I think it is. I think it's true especially in areas like privacy, for example, which don't really require a heritage of five years of deep knowledge of write-ahead logging or Mohan's 27 concurrency modes and techniques in the Aries arena.

One persistent cornerstone of relational database research has been the belief that the query optimizer should be able to come up with a good plan for optimizing any *query. But the industry's concept of user-directed query optimization seems to imply that the user can't be removed from the loop. Does that mean that the research community has failed to deliver on its promises? Is more research needed here?*

In my view, the query optimizer was the first attempt at what we call autonomic computing or self-managing, self-timing technology. Query optimizers have been 25 years in development, with enhancements of the cost-based query model and the optimization that goes with it, and a richer and richer variety of execution techniques that the optimizer chooses from. We just have to keep working on this. It's a never-ending quest for an increasingly better model and repertoire of optimization and execution techniques. So the more the model can predict what's really happening in the data and how the data is really organized, the closer and closer we will come [to the ideal system].

So, let's take a look at this notion of hints to the optimizer. The problem with hints, particularly now, is that people can't type in an ad hoc query and have a database administrator run into the room, crying "Don't hit 'enter' yet, don't type that last semi-colon, let me put the hints into your query before you hit the 'go' button." Packaged applications of PeopleSoft, SAP, Seibel Systems---just go down the list of applications: they're all pre-packaged. You can't get at those queries, you can't put in hints about your arrangement of the data. So we're forced into a position where for an ever-increasing set of queries, there's no way to have user-defined query input or hints or whatever you want to call them. There's no way to do it. So, we've got to keep on investing in that technology.

Is there some way that maybe the hints could be passed up to these pre-compiled packages so they could deal with the different situations on the fly?

Well, there are always ways you could have the software vendors themselves put the hints in: "If Oracle then …; if DB2, then …; and so forth. I don't see them being really willing to do that. As an independent software vendor, I wouldn't want to have to learn what performs best in what query context. And I don't know how big the sales and distribution tables are going to be, how big the HR tables are going to be.l

What I was imagining at run-time, for example, was that hints like the statistics or the physical layout information could be passed to the package, and then the package could be compiled in such a way that the right plan would run to match---

That is what actually happens. We look at the statistics on the data within the engine catalogue and that's what gets used to do the cost-based query optimization. So, to the extent that that model is working properly, you already have hints.

> [When I managed Jim Gray and Bruce Lindsay,] we all had long blonde hair. So there was no diversity there.

Ah, I was imagining that the packages already had the queries compiled and ready to go and you had no flexibility and they were going to run a certain way no matter what.

No, they don't come from the factory prewired in that manner. They come from the factory saying, "execute this string." And then the optimization happens based on the statistics on the data for your particular installation.

I came across a picture of you from back in the early days of System R where your character seems to shine right through and at the same time you clearly do not at all fit the stereotype of a 1970s computer science researcher, no greasy hair, no slide rule, no pocket protector, no glasses even. Over the years, has it gotten easier or harder for people who don't match the evolving stereotype to fit in?

I think that---gosh, I wish I still looked like that---people are more accepting of diversity today than ever before. In fact we look for it in many ways because we want fresh infusions of ideas. So I look for people who have backgrounds in something outside of databases very often because I want to bring the richness of these other experiences to bear. I think that carries over to diversity in lifestyles and world views and cultural background and those kinds of things.

You know, I haven't thought of it this way before, but now that you mention that they gave you Jim Gray and Bruce Lindsay to manage at first---I've seen pictures of them back in those days and it almost seems a bit cruel to hand them over to this clean, fluffy creature.

This young girl, yeah. Well, we actually all matched. We all had long blonde hair.

Alright, okay.

So there was no diversity there. [laughter]

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

You know, I always wish that I had more time to read what else is going on in other areas of the database field and beyond. There's never enough time to do what I wish I could do.

The situation seems to get worse as we have more conferences and more database researchers and more interesting things to read too.

Right, right. Absolutely.

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

I'd just like to know more about hardware, I think, that would be---

[Surprised.] *Hardware?*

Yeah. I think that would be an area that I would like to learn more about.

How would that affect what you did? How would that affect your thinking?

I think right now every database is faced with this world where the processors have gotten twice as fast and the disk has gotten maybe four or five or six percent faster. So all of a sudden there's this huge, huge gap in performance. I wish there were ways that we could fill this gap by avoiding I/O, overlapping I/Os, prefetching into the buffer pool. We have a lot of techniques that we continue to invent to cover up this I/O performance gap. I think if we knew more about hardware and could spend a lot more time with the processor I/O set of people, storage subsystem people, and so on, there might be some more invention that we could do here that would benefit everybody.

You know, one thing I think is odd is that if you take a fresh database PhD, the chances are almost zero that they will know in detail how a disk works.

Yes.

There's that paper by John Wilkes that describes a disk in detail, and I didn't know the things in there even after being in the field ten or fifteen years by the time I read it. Maybe there's some kind of gap in our education of database people that we don't learn what's under that hood.

Right, right, and as the field grows, more and more of that gets abstracted out. Today we have this sort of simple-minded model that a disk is one arm on one platter and [it holds the whole database]. And in fact [what's holding the database] is RAID arrays, it's storage area networks, it's all kinds of different architectures underneath that hood, and it's all masked over by a logical volume manager written by operating system people who may or may not know anything about databases. Some of that transparency is really good because it makes us more productive and they just take care of the details. Here I am trying to make self-managing database systems, so [I should be happy for them to] make self-managing file systems too. But on the other hand, optimizing the entire stack would be even better. So, we [in the two fields] need to talk, but on the other hand we want to accept some of the things that they're willing to do for us.

Thank you very much, Pat.

Oh, you're very welcome. It's a pleasure.