Laura Haas Speaks Out

on managing teams versus children, research versus product development, and much more

by Marianne Winslett and Vanessa Braganholo

Laura Haas

https://researcher.ibm.com/researcher/view.php?person=almaden-laura

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 Indianapolis, site of the 2010 SIGMOD and PODs conference. I have here with me Laura Haas, who is the Director of Computer Science at IBM Almaden Research Center. Laura is an ACM Fellow, an IBM Fellow, a recipient of the SIGMOD Contributions Award, and a member of the National Academy of Engineering. Her PhD is from the University of Texas at Austin. So, Laura, welcome!

In the US, NAE membership is the engineering field's greatest honor. Is it true that you did the work that led to your induction because you wanted to get out of creating a version of R for PCs?*

Yes, in some sense it is! We were asked to build a version of the relational database that would run on a PC, back in the days when PCs had 640K of memory, and very limited space, and all these tight constraints. It was all architectural, and understanding the hardware, and the bits and

the bytes, and I've never been particularly interested in the low level details like that. I was always interested in the abstraction layer. So we were kind of looking around for what else we could do while we were doing this. Then we decided we were going to have to create a very limited system if it was going to run in 640K, so it would be really important if it were extensible to new sorts of things, and new function and new types of data. So, in some sense, that's true, I would never have phrased it that way perhaps, but yes!

What did you do instead?

We created Starburst, which was one of the first extensible database systems. It came up with a new way of optimizing queries, it's very extensible, rule-based, elegant (I think elegant, Guy Lohman did the work, I should be very clear on that!). But it was a system that would allow you to plug in new data storage systems, that would allow you to add new indexing methods, that would allow you to add new functions, one of the early user-defined functions capability, and still be able to optimize over these very different kinds of structures. Somewhere in there, the product division decided they didn't really need that much help to create their little PC database, which was fine by us, and so we got some space to do some real innovation, and to really rebuild from scratch, a second generation relational system. I guess what kind of put that work over the top was that it became the basis for IBM's DB2 for workstations (the DB2 that most people would know of and use today, not the mainframe system). To this day, some of the code we wrote in that project is part of that system.

Your career has certainly turned out well, considering that I heard that you were forced to go into the database field. Why the reluctance?

I had a really bad experience in grad school -- I'm old enough that relational databases were being invented when I was in graduate school. When I went to grad school, database was these network database systems, these complicated models, and we had a teacher who shall remain nameless, who just did not make it terrifically exciting. I didn't like the class at all. I declined to take my database qualifier because I was pretty sure I would flunk, because I was putting no effort into the class. So I was somewhat horrified when I got to IBM and they told me my first project would be a distributed database project. There had been a highly available systems project I thought was more interesting, but, in retrospect, I was very lucky, because that project didn't go, I think, as far, but it was doing the kinds of systemic and algorithmic things that I liked with distributed systems.

So you didn't do databases at Harvard with Phil Bernstein? Cause that was 100% relational…

No, I did not. I took his operating systems class, I took compilers… I decided very late that I wanted my minor, because at that point it was an applied mathematics major, in computer science, and so I didn't get as far into the curriculum as some people did. I spent a lot of time thinking I was going to major in linguistics, and in fact at Harvard, I got them to offer a seminar in Natural Language processing because I was so interested in languages and grammars and so

on, and I knew people were doing that with computers and I wanted to know what was what. So I asked my advisor if there was any work in that at Harvard, and he said "no, but you know Tom Cheatham (who was the head of the department at the time) is really interested in that topic. Go talk to him. Maybe he will run a seminar". And they did, they did run a seminar and it was an all graduate seminar, and me. So I did a lot of other things and never made it to databases, it is sort of a shame. I did audit the database course when Umeshwar Dayal got to Texas and started teaching it, and it was so night and day different from what I had seen, that I really wished that I had spent a little more energy, and learned a little more. But no, I didn't intend to have anything to do with databases ever again.

You have spent most of your career in IBM research, but you also spent time on the products side at IBM. What should researchers know about product development?

I went into the product side as an escape. I had gotten myself too over-committed to all the things that were going on in my little industrial research world. At the time that I went there, I was transferring the work from Garlic into the product division. I had just started the Clio project, which was, I think, a fairly radically new, at the time, approach to do information integration, from the tooling side, not the infrastructure side and how we connect up different components. It was a cool research project. We were doing really great stuff, building prototypes, and writing papers, and at the same time, IBM had just launched a life sciences business, and they had decided that information integration, in particular the kind of federation of heterogeneous sources that Garlic had specialized in, were really key to getting a foothold in healthcare, in particular, the pharmaceutical industries. I found myself CTO of the life sciences business for IBM, in my spare time, while running this research project, and transferring my old one, and I was going nuts. I call those my guilt years. I had a young teenage son, and one in middle school, and they needed a lot of time and attention. So I was always a bad mommy; a bad researcher, because there would be a paper deadline, and I was talking to clients; a bad CTO because I should be on the phone with a client, but I'm writing my paper; I was always bad to somebody. So I went to development as an escape. I somehow just decided that there was no way to fix my life except to give up one or more of these roles; I just had to give it all up. And what I think database researchers should know is development is no escape! The pressures were, if anything, even more intense, there was a huge learning curve. I was Little Miss Purist Researcher. In fact, when I told people at IBM that I would be going down to development as a development manager, they were like: "You? That doesn't make any sense! You're the one who has had the least to do with development!". I was never hostile, but there were always people to buffer me, and I just didn't engage very well. So I learned more acronyms in a month than I had learned in my previous 20 years at IBM. I learned a lot of really cool things about how you get a product out the door, and all the work that goes on between what we in research think is a finished product, the cool prototype that we could write some new papers and run some great experiments on, and what customers really use. I learned a lot about how you have to support customers, and deal with customers. So I would tell people it is a really worthwhile experience to

have, but don't do it just for the sake of being in development. Do it in something, an area you are kind of passionate about, pick your timing perfectly. I was really lucky. I went down there (it's the development "down there" because IBM Silicon Valley lab, our development lab, is 5 miles south of Almaden, and we're on the hill and they are on the flats), so I went down there to be the development manager for the query compiler for DB2, which was like going home to your grown-up child. So remember that Starburst code became the product basis, and it had grown up over the 10 years since then, or whatever, and so now it was a full-fledged product offering. It was kind of cool to be invited back to actually manage that team, and it gave me a certain amount of stature with the team, which was good since I didn't know anything, and I relied desperately on all the managers and people around to get me through. But the timing was perfect because, in fact, this group also owned the technology that we were transferring from Garlic, and they went on to launch our information integration business, and I got to be the development manager for this new business.

Why should one choose to become a manager?

I think being a manager gives you a very different perspective on the world in an industrial lab. It is kind of a growing up experience. It forces you. Some people do this naturally. But for me, at least, it really forced me to think about not just other people, and careers, in a way that I would have never thought about my own, but also to think about where the work had to go in developing a vision, to provide that kind of guidance and direction to people, and to help them understand. I don't ask a lot of questions of myself, I just kind of blindly do things, and so I think the thing about being a manager is other people asked me the questions that I should have been asking myself.

Like what?

Like, "how does this have anything to do with IBM's bottom line?", "why will anyone care?". Oh, that's an interesting question! "Ok, so what is our value proposition"? And then, I'm a smart enough person. Once I've been asked the right question, and I'm thinking about it, I can start to develop answers that I think are compelling to me and to others. But at least early in my career, I didn't have that ability to ask the right questions, now I do. Now I've learned what a lot of those right questions are.

What are some other ones, besides that one you have already mentioned?

Things like: "why does it have to be this way?". I get frustrated with procedures and processes and I'll ask about those myself. But, the more meaty questions of, "why do we have these kinds of rules?", and "why is it important that I join ACM?". I mean, I joined ACM as a student because my advisor told me to. I really had to think about: "well, okay, why did I stay a member of ACM?". We know this is a huge issue for ACM now, and now I can at least talk to people.

So, what is your answer?

I think you do it for your career. You can just show up at the conferences, you can just read the publications, but when you join something, you get a certain responsibility for it, and that kind of professional commitment causes you to change your behavior. I think if I had not joined, I would not have understood why I should be a vice president of SIGMOD at some point in the process. That was a wonderful experience, for my career, for networking, for understanding how things work in a broader world than IBM, and so on. There is a whole range of things. You don't do it for the publications; you really do it for your own career. I'm kind of a stickler for that. It is one of the things I lecture my team on. It doesn't have to be ACM, but some professional society, something outside of your company, or your university, or whatever. I think it is really important for your profession.

And in the part about questioning rules and procedures, do you mean you should have been more often questioning them, or more often accepting them?

I should have probably asked myself more questions. You know, somebody would say, distributed database (my first project at IBM), homogeneous database, "that's a good project", and so I just kind of worked on it. When we got to the second project, and they made me a manager, and they said we want this database for workstations, fortunately people in my group said "where is the research?", "what is it that we are doing?", and that was really what forced me on this path of, what are the key questions for us, and how are we going to answer them, and into this extensibility thing that we talked about. I was a shy, timid sort of person, and so for me, it was important just to grow up. People don't believe that now, that I was ever shy and quiet, but it was true. And I think even for people who do ask lots and lots of questions, I've seen it have almost the opposite effect: it forces them to kind of think about how what they say is interpreted in a different way. Because, as an individual, you are only speaking for an individual, and if you screw up, and somebody's mad at you, then that is your problem. If you are speaking for a team, and you didn't think before you spoke, and somebody decides that your project is crud, this can have a big impact on people. So, you get a level of responsibility that's very different. The other reason I always tell people they should be managers is because it is really good for parenthood, and vice versa. I learned a lot about being a parent from being a manager, I was a manager first, for about 8 months, I think. I took my first management job when I was pregnant (not sure I knew that at the time). So I had 8 months of practice, and you know, all the annoying little questions people will ask you, and how you have to teach people to think for themselves, and so on… When you go and apply that to children, it teaches you a lot of patience. I thought it was very useful.

So it sounds like it can help our readers in a lot of ways. So what sort of annoying little questions does one need to be ready to handle?

People think you are an instant expert on process and procedure. I am the worst person to ask about a process or procedure. I've learned to teach people to find their answers themselves. I teach them who my resources are. Who would go and do these things. It is hard to make

sweeping generalizations, although I love to do that. But there is a class of people that are just very dependent, like the young child who doesn't want to let go of mommy's apron strings. They just ask mother-may-I way too often, and you need to kind of teach those people just like you teach a child that "you are responsible, you can move out from here". But that clinging… all the mothers know the clinging child that holds your apron string. Then there are the others that just cause trouble without thinking. They just speak without thinking, they offend somebody they really didn't have to offend, probably a teammate. It is like two children squabbling. You can just take any of the scenarios of parenthood, and they play out in the workplace, in a much more refined way. I mean, clearly, people don't punch or bite or scratch, but you get all the variants of behavior, which is why the analogy to parenthood works. I also found that it worked the other way: that being a parent taught me patience that I didn't have for grownups at the beginning. You get used to with your own children, to having to bite your tongue, and then think how to get them to do what they need to do, and that works well for the grownups too.

So thinking back, we have a lot of readers who are young people from distant lands, so some of them will be shy, and retiring. So thinking back to yourself all those years ago, what would you say to that person, from the vantage point where you are now?

I would say that too often people are afraid to try, when they offer us a position that we think we can't do, one that sounds like a really big stretch. Often they have seen something in us that we might see, but are afraid to admit to in ourselves, and we should just go with that. You should never take a position unless you really investigate it and you think you will enjoy it, and get something out of it. You should always take a position if you think it sounds like fun, but it is really scary. You know, it probably means you have ideas, and it probably means you are going to grow a lot out of it. So, it is hard for people, I know it was hard for me, when I was really young to think that I would ever enjoy being the leader. And I love it. I grew up into that. It took a long time, but I grew into it. And I think a lot of the readers, if they give themselves a chance, will too. So it is an experience I recommend. It doesn't have to be a manager, it could be a professor, you know, professors do this. It could be just a technical leader, leading some taskforce. That is really scary. It could be vice chair of SIGMOD or chair of SIGMOD, who knows, right? But they should give themselves the opportunity if people start to push them into it, even if to them it sounds like "wow, why would they think of me?". Especially, I'd say that to the women, and some of the folks in the global community, for whom it is just not natural to put yourself, as we would say in the States, "out there". Try it.

What about the reverse? In your management career, you have probably seen people who are naturally very bold, so we told the timid ones to be bolder, maybe push your comfort zone after investigating, and people may see things in you that you don't see in yourself. What about the people who are naturally very bold and want to lead everything? Do you have any words of advice for them when they are young?

Yes, learn to listen. Because, even if you are the most brilliant thing out there (and you may well be, I know some people like that), there are people around you who will have a different perspective, and who will teach you things, and make you even better and even more effective if you listen to them. What I often find in the brash young folks, is they are so busy impressing people that they don't necessarily listen and catch the subtleties of what people might be trying to tell them. So it is good to practice that if they are in a management or leadership position, professor, or whatever, they should practice taking a certain amount of time at each meeting to ask questions and then just "shush" and listen, because you need to get those skills. People tell you things if they think you will listen. If you aren't good at listening, you will lose that.

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

Enjoy. I just think this is the perfect moment to be in this field. I told the new researcher's symposium earlier this week that I am very, very bullish on the field right now. I see us poised to deal with all these vast data volumes, and different types of information. We have been promised for eons (it feels like) that we were entering the information age, right? We all believe we are there now, but we are still not getting all the value we should from that information. We are at the tip of the iceberg in terms of the impact that all this information is going to have. I think in the future everything is going to be about making all our processes and systems smarter: smarter energy, smarter health care, more individualized, less wasteful, all these different things. It is all about analyzing information, and then feeding that back in a control loop, and we can be at the heart of that if we want to. If we can understand what it is people want to do with the information, we can really pay attention to those applications, and engage to create the solutions that are needed. So I think it is just a really exciting time. We are moving from where I started, "oh, we are going to build some repositories for information, isn't that nice?". And now we are finally in an era where we can control the information and be the player that makes all this value come to pass from that data. So enjoy it. I think it is ours to really win and have a huge impact on the world. I am very bullish about that. I don't know another way to say it.

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

Yes, I do. Probably my favorite piece of work is the set of work Renee Miller and I started on something we called schema mapping. In the late 90's we had this notion that all the work that had been done on integrating data up to that point looked at what I would call the plumbing, the pipes, and how do you connect up the pipes to get data to flow to a particular place. But nobody had really thought about the design much, and how do you specify were those pipes should go, and what should happen to them, and so on. We set out to build some tools that would help people say how data should be transformed, and then we just applied database technology, especially the notion of having a higher level of abstraction and non-procedural, and all these good words we learned in the relational era, to the problem of information integration and ended up defining a new notion of schema mapping that would map from one schema and would say

how one schema was related to another in a non-procedural way. Many people had talked about that before, but what we realized was that you could pose that problem in a way that could be viewed as query discovery, where you were really trying to find a query that related the schema you wanted to see with the schemas that you have. And if you could view it as a query, then you could compile it into many, many different runtime platforms. So this became a project called Clio, which started about 11 years ago, roughly, and wrote the first paper for VLDB 2000. So really the idea started to form crisply 10 years ago. I just really like that work because I think it is very elegant. It spawned a little subfield, at least for the theory community. There has been a lot of follow-on work that's happened, and I just feel very proud of it. I am very pleased with that piece of work. I think there is a lot more that needs to be done, so I am really glad other people have taken on some of these challenges, and are moving way past what we thought about. But for me, it was just a very nice elegant idea that turned out to be very practical. Some of the theoretical work led us develop algorithms that were very efficient for generating the transformation code, and then the ability to compile to multiple platforms such as federated database, so you could compile it to SQL, and then the federated database could execute it, or you could compile it to ETL scripts, or XPath and XSD, and a lot of XML transformation tools, or even to Java code. That was very powerful, it meant that you could express what your data looked like today, and what you wanted it to look like, and then depending on what a user's preferences were, you should be able to generate code to run on the right low level plumbing to meet the requirements of that user case.

What did people do before that work? They still had databases with different schemas before...

As I said, we had lots of plumbing, so they would make a decision: "I'm going to use a federated database system". They would then sit down and write by hand a bunch of SQL queries that would have to be debugged, and so on and so forth, that would in essence be the views that would transform the data. So it was a very manual process, and you weren't really saying how the data were related to each other, you were writing a query. It is just like writing code instead of writing the SQL query, only we were trying to move up another level in abstraction. Or they would say "I'm going to build a warehouse, so I am going to use one of these extract/transform/load ETL tools". And depending on which tool they chose, they would either write a little script (really write a program to extract data from one place, and put it to another), or the more sophisticated ones gave them these lovely dataflow tools where you could draw, you could just put things on a canvas and connect up the operators, and so on. It is still a program. What had spawned the Clio thought was my experiences in the product division, working with customers, and seeing how they would often like to use a federated database to prototype, just to see what they could do if they brought the data together, because they were much faster to set up. But the performance, if you were dealing with large data volumes, or doing these sorts of queries over and over again, it was not so good as if you built yourself a warehouse. Warehouse projects could take 6-12 months. They didn't want to go in there until they knew they wanted to do it. So they would set up a federated database, and then they'd ask, "so we think we understand what we

want to get over here, but we have to build a warehouse. Is there some magic way to do that?". And there wasn't a magic way. I think what made Clio different was it offered a magic way: "well, you told us what the relationships were, and first we generated the federated queries, and we got your federated system working, and you got to see what you liked, and you could switch your mappings and regenerate the queries, and you could iterate much more quickly, and now when you are happy, you could generate out an ETL script and it could build your warehouse for you". That's cool :) We had a lot of success. In IBM at least, the Clio code has gone into a number of different IBM products, and so that is really nice as well. That is why that is my favorite work.

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

This will be difficult to explain. I would be writing a newsletter. Because I have this worldwide role in addition to my day job running the computer science department at Almaden where I'm trying to help IBM's worldwide research team understand what it means to do research. I have a lot of young labs with very young researchers, some in economies where there is not quite the same tradition of scholarship or certainly of industrial research that we have. So I'm sort of team mom for the global team in terms of helping ensure that we have really shining cool projects and scientific examples going on worldwide, but also that the researchers grow and learn and so on. And so I feel a desperate need to communicate more with the global team. I can't be everywhere at once, unfortunately. I think a newsletter would help, but it takes a long time to write a newsletter, and I just don't have that kind of time.

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

I wish I had Mohan's ability to remember every article I ever read, and person I ever met, and how they are related, and who knows who. That would be really wonderful. And I wish I were more detail-oriented. I have to force myself to focus on the last, you know, crossing the T's and dotting the I's is not my strength. I am one of the people who would paint with a broad brush and wouldn't be so good at going back and filling in the details.

But when you are high up, you can't fill in the details, can you?

I know, but I think you could do better. I think I could be more disciplined. But I think it would be more useful for me to have that Mohan encyclopedia knowledge. That would be nice.

Thank you very much for talking with us today.

My pleasure.