Kenneth Ross Speaks Out on Making Contributions that Span Technology and Science

Marianne Winslett and Vanessa Braganholo

Kenneth Ross

http://www.cs.columbia.edu/~kar/

Welcome to ACM SIGMOD Record's series of interviews with distinguished members of the database community. I'm Marianne Winslett, and today we are in Snowbird, Utah, USA, site of the 2014 SIGMOD and PODS conference. I have here with me Ken Ross, who is a professor at Columbia University. He was a Sloan Fellow and a Packard Fellow, as well as an NSF Young Investigator. His Ph.D. is from Stanford.

So, Ken, welcome!

What is next for main memory databases?

So, I think as we go forward, main memory databases are going to be the mainstream databases. We're going to be thinking of disk-resident databases as secondary. The primary copy of the data and the primary activity is going to be in main memory. IO and so on are going to be things you think about just for recovery and persistence. So main memory databases are the mainstream. The obvious questions are how to get very fast transaction processing, very fast analytics. As my research interests kind of reflect, the hardware is evolving relatively rapidly – you're getting multi-core machines, and on these multi-core machines, you're getting various kinds of hardware capabilities. I think the critical thing going forward is making the best use of these hardware capabilities. Things like SIMD units, things like transactional memory, gather instructions, relatively low-level things but they can make a very significant performance difference when they're in the inner loop of a database join, or aggregation or some important operation that is run many billions of times.

I think the Database field is particularly good, perhaps better than the many other fields in and out of Computer Science, in admitting work that goes all the way from theory to systems.

I can see how that would be really important, but isn't the rate of data collection expanding faster than memory size?

The rate of data collection is growing very rapidly. People are trying to scale systems accordingly. The RAM sizes you see are dramatically increasing. That being said, some of the biggest customers of Oracle, for example, still have datasets that reside in large main memories. Oracle is actually supplying main memory systems to those customers. So, while there will always be applications on the fringe that are in excess of what we can store in main memory, things like large astronomy datasets and so on, many of the applications, a lot of the critical ones for economic or scientific analysis will have working sets that do fit in main memory and can benefit from these sorts of technologies.

For our younger readers, can you say a few words about how being in main memory changes everything compared to the old disk-based days?

Okay, so in the disk-based days, you would have to wait for an IO to do most operations. An IO would take a few milliseconds for the data to come in and if you had to read that disk page to find out what the next item you need to read is, then you have to wait for a second IO that would have to be in sequence, you can't make them concurrent. So, you have a lot of latency. Once the data is in main memory, you are now thinking about nanoseconds rather than milliseconds, so you have six orders of magnitude potential difference in speed to access data. Then you start caring not about whether it's on disk or in memory but is it in memory or is it in the cache? The caches can be a factor of 50 to 100 faster than the main memory in terms of access. You have similar problems in terms of trying (from a disk-based database) to buffer the disk-resident data in RAM – you see analogous problems at high levels of the memory hierarchy trying to put the data you need not in RAM, but in the cache (at least for a short period), to take advantage of temporal locality and get faster performance.

We're still teaching them in the courses about B-trees. Everything is based on B-trees. Are we teaching them the right thing?

In my database implementation class, we start out teaching them about B-trees, so I think for historical reasons, it's important to be founded in that kind of knowledge, but then we go on and talk about cache sensitive B+ trees[1] which is actually something I worked on with my student Jun Rao back in 2000. You take the basic B-tree structure and you re-work it to make it work well in main memory. So instead of having k+1 pointers and k data items, you'll have 2k data items and 1 pointer. So, you can fit much more in a node, and you size the node to be the size of a cache line. You get away with this because the pointer now points to a contiguous set of child nodes and so as a result, you can use arithmetic to figure out where the child node is. Also, you get a much higher branching factor for one cache line, so you get fewer cache misses during a

[1] Jun Rao, Kenneth A. Ross: Making B+-Trees Cache Conscious in Main Memory. SIGMOD Conference 2000: 475-486.

traversal. So, the underlying concept of a B-tree is still there, but as a research community, we have evolved it to suit the appropriate kinds of memory technologies that are appropriate for the time. So, I would hope, and I do get told by some of my colleagues at other institutions that things like cache sensitive B+ trees are now being taught in the mainstream database classes.

Excellent. Is your research valuable to industry?

I think it's very valuable to industry. I have some collaboration with companies like Oracle where they're taking interns from my group. Some of the problems that we're working on, for example, there is a paper[2] that we have at SIGMOD 2014 on track joins that is motivated by a problem at Oracle where you have very large joins over a cluster of network machines. My student Orestis and I have come up with a technique to do joins kind of like a semi-join where on a key-by-key basis, the data is re-partitioned which ends up transferring a lot less data than say if you did hashing. This is a critical workload for Oracle. This is the slowest query in an important customer workload that we were able to speed up significantly. So that is just one example of where these research techniques can have a fairly direct impact.

To have that impact you had to know that it was the slowest query for an important customer. So how do you build that relationship where you learn those types of things?

I think in this particular example, Orestis, my student, was the key player. He was there as an intern. He found out about what was going on in Oracle. I don't take all the credit, but I perhaps can take credit for placing Orestis in Oracle and working with Eric Sedlar, for example, to find a project where Orestis's strengths could be most utilized. It's been fruitful, and we have an ongoing collaboration with Oracle.

Your Ph.D. work was entirely theoretical, but most of your subsequent research was on the system side. Do you have tips for making that transition?

It was a complex thought process I had to go through as I joined Columbia as a junior faculty member. I did a thesis in which I developed the well-founded semantics for Datalog[3,4] and that was actually relatively influential. It had a number of citations and formed the basis for a lot of work in Datalog. So, I was sort of branded as being the person, along with my collaborators John Schlipf and Allen Van Gelder, who developed the well-founded semantics. And this is a blessing and a curse in a way because as I progressed in my career at Columbia, Datalog kind of went out of fashion. If you're working in a field that kind of goes out of fashion and people point at you and say, "Oh he did the important Datalog thing," even if you are doing other things, that's what they remember you for. So, it takes some effort to actually take what you're doing next and make it known in the community. In the period before I was coming up for tenure for example, I went to various other institutions and labs and gave talks about some of the work I was doing on query processing and optimization and so on which was more applied and it was sort of hitting much more in the direction in which I find myself right now.

Do you think that a young person starting their first tenure-track job today could make that big switch to systems where you had to really build things before you can publish – would they have time to make such a big transition before tenure?

That's a tricky question. I think the best students are able to span theoretical and practical concerns. I think the Database field is particularly good, perhaps better than the many other fields in and out of Computer Science, in admitting work that goes all the way from theory to systems. Even though I have moved from theory to systems over the years, it just means I used to publish in PODS, and now I publish in SIGMOD and VLDB more, but it's still the same conferences, I still circulate with the same people, and I think that interaction is good.

So, coming back to your question, I think there are theoretical people who prefer to work on purely theoretical problems, and that's fine, but if you're a theoretical person who has an inclination to write code and implement things like I like to do, I think it's fun to play with that. Don't necessarily invest all of your energy in that. Have one or perhaps two side projects that may or may not pan out. Keep your mainstream work that you feel you have the most cutting-edge advantage in your research going, but do these side

[2] Orestis Polychroniou, Rajkumar Sen, Kenneth A. Ross: Track join: distributed joins with minimal network traffic. SIGMOD Conference 2014: 1483-1494.

[3] Allen Van Gelder, Kenneth A. Ross, John S. Schlipf: Unfounded Sets and Well-Founded Semantics for General Logic Programs. PODS 1988: 221-230.

[4] Allen Van Gelder, Kenneth A. Ross, John S. Schlipf: The Well-Founded Semantics for General Logic Programs. J. ACM 38(3): 620-650 (1991).

projects. These side projects can often expand and become products that take a life of their own, and they can drive you in these new directions. That was kind of what happened for me.

In some sense, you've come full circle because your recent sabbatical at LogicBlox involved working with Datalog. Has the time finally come for Datalog in our field?

That's a very interesting question. I did get approached by the principal people at LogicBlox, including Molham Aref, and they sort of looked at me as if I was this really famous rock star type character because I had done well-founded semantics. And here I am 20 years later, having put that in my past and not being used to people thinking of me in that way. I have to say it was flattering. I enjoyed the attention from having that kind of feedback. Then it led to some interactions, and as a result, I did go to LogicBlox and did some work while I was there related to some of the interesting problems they were having, that overlapped with the research I had done in the past. So that kind of recapitulated and I looked at it in a new way that might be relevant to LogicBlox. And even after my sabbatical, I've managed to continue having a consulting relationship to LogicBlox that I think is helpful for both sides.

Coming back to your question about if this is the time for Datalog… what I really like about Datalog is its declarativeness. I think that SQL has succeeded in the relational database community because it's declarative. People don't know how to program yet they can write SQL queries, so it takes less effort, energy, and knowledge to master that technology. Datalog has the disadvantage that it is a logic language and people are often not as inclined to think in a logical framework in terms of predicate calculus and so on. On the other hand, sometimes you can use syntactic sugar to hide some of those complexities. With Datalog, you can use recursion to express things declaratively. Some of the work for example, by Joe Hellerstein in Berkeley is using recursion to reason about time and protocols and so on. I think that's an excellent kind of direction because it's taking advantage of the declarativeness, but using a fairly limited expressive power language to write your specifications so that you can reason about them, prove correctness results, and form a layer of abstraction that is much cleaner than an arbitrary procedural code. So, I think it's cool that these additional applications that weren't really foreseen for Datalog have come along and are making it relevant again.

You are unusual among computer scientists in bringing a broad scientific perspective to your work. In fact, most people don't know that you've co-authored published articles in physics.

Yes, when I was an undergraduate student, as a summer project I worked with an applied mathematician who worked in physics on a model of the Ising spin chain. I don't actually understand it in full detail, but I did some coding of some physical simulations that corresponded to the physical problem that he was studying. It turned out that the results were kind of interesting. They showed a fractal structure that was somewhat new, and as a result, we got a couple of publications in theoretical physics journals.

I like the idea of working in the scientific field itself, trying to understand the domain rather than just building a tool to help the domain scientists and I think that provides a much more satisfying and rounded type of experience in making a contribution.

And you're still working in science, although more recently it's been bio-informatics. How is it to work with bio people?

Some of this interest in biology came from a point where the human genome was about to be sequenced. The various universities were being called to help work in the sequencing of the human genome, and some people from the Medical School at Columbia came down to the Engineering School to try to recruit people to work on the sequencing effort. In order to get them interested, they gave a little short course on biology, a five-lecture sequence in which they taught basic biology to engineers. So, I attended this course, and I was actually fascinated by some of the biology. I didn't get involved in the sequencing effort at the time, but it got me thinking about many of these questions of biology.

Over the years, I've actually worked on a couple of research questions in biology sort of on my own as one of these pet projects as I've mentioned before. For

example, you might not know I wrote an article[5] about why some groups of species have a very variable chromosome number among different species in a clade while other groups of species have a very conserved number of chromosomes within a clade.

More recently, I've been thinking about autoimmune diseases, and I have an article studying the genetics of autoimmune disease with the hypothesis that the cause of autoimmunity is an immune response against mutated genes (mutated proteins that are expressed in the body). In order to explore this, I took the human referenced genome and ran some SQL queries on these referenced genomes from the UCSC database[6] and found a statistically significant overrepresentation of genes with very long repeat regions among auto antigens. This was kind of exciting, and I wrote it up. It required a lot of reading and a lot of understanding of the biological literature. It appeared at PLoS One[7].

That work I did on my own, but in the biology field, people don't really take you seriously until you've got an experimental validation of your ideas. I talked with one of my colleagues who does computational biology, and he recommended I speak to a certain person who studies inflammatory valve disease at the Mount Sinai Hospital. Her name is Judy Cho and so I'm working with her and some other people at Mount Sinai Hospital to experimentally validate this hypothesis. So, in this particular case, I've worked somewhat on my own but then done the collaboration afterward.

Coming up for tenure, it's important to have your work known by the community. Give talks about your best work and visit other labs.

I like the idea of working in the scientific field itself, trying to understand the domain rather than just building a tool to help the domain scientists and I think that provides a much more satisfying and rounded type of experience in making a contribution. Some of this work was inspired by my Packard Foundation Fellowship. You did mention I was a Packard Fellow (from 1993-1998) and one of the things that the Packard Fellowship does is that it brings all of the Packard fellows together and they give talks about their work. Just as an aside, one of my fondest memories was getting a pat on the back from David Packard as I gave my talk at the Packard Fellows meeting. And so, I'm up there giving my talk, at the time I was doing some theoretical work on object-oriented databases, and these other scientists were talking about cures for Malaria and various other very high impact things. I was kind of scratching my head thinking, "Here I'm doing object-oriented database theory, and these people are really impacting the world." I kind of had this urge; I want to impact the world too. I stayed in my main area of Computer Science, and I still worked on databases, but I had a strong incentive to do one or two of these pet projects to try to explore things outside that domain. I just followed my curiosity and had fun, so that's how this biology project eventuated.

It sounds amazing! Stepping back for a moment to the validation, is that going to be more SQL queries over particular patients' genome or is this stuff they're going to do in a wet lab?

These will be wet lab experiments. There's a particular technology that allows you to sequence genomes in particular regions with fairly long reads. That will enable you to look for certain structures that should, according to the hypothesis, differ between patients and controls. These structures are actually not easy to detect with current technologies because they're longer than the read length that most of these short read technologies give. So, the nice thing about this collaboration at Mount Sinai Hospital is that they have this database of 30,000 patient's blood samples that they can go to, and you can get 100 people with a certain disease and get their blood samples and test them versus controls at relatively low overhead. It takes some effort to setup the scientific experiment, and there are all kinds of design issues for the experiment that are things that I wouldn't have thought of at first, but my collaborators there have to go through to make sure that the experiment is going to succeed and find the things we're looking for. That's where it's essential to have collaborations because I have no wet lab experience and we need to bring out our respective strengths to be able to solve these bigger problems.

By thinking like a computer scientist, you came up with this hypothesis for autoimmune disease. Does that mean you have a hypothesis for how to cure them?

If this is, in fact, the mechanism that causes autoimmune disease, which is speculative at this point because it's just a statistical association, it's not validated, so I do

[5] Kenneth A. Ross: Alpha radiation is a major germ-line mutagen over evolutionary timescales. Evolutionary Ecology Research, 2006, 8: 1013–1028.

[6] https://genome.ucsc.edu/

[7] Kenneth A. Ross: Coherent Somatic Mutation in Autoimmune Disease. PLoS One. 2014 Jul 2; 9(7): e101093.

not want to claim that this is the solution to autoimmune disease. But let's imagine for a moment that in fact, somatic mutation of these proteins is what triggers autoimmune disease. It opens up certain possibilities. If you know the specific proteins that might be causing the autoimmune disease that have been mutated in a way that is relatively deterministic and predictable, you could do various things relative to that particular protein. You can try to induce tolerance to that protein, for example, or you could find ways to take that protein out of circulation one way or another. My knowledge of biology is limited in terms of knowing the different options for which you might use that knowledge, but if you know the basic procedures and steps that trigger a disease, you can go early in this triggering process, identify the early players and try to get things as close to the causative part of the mechanism as possible. So, by extending the knowledge base and by making the knowledge closer to the triggering point and making the identification of very specific targets, I think that opens up much more opportunity compared to alternatives like just generally dampening the immune system, which can be effective and is the current treatment for many autoimmune diseases but is non-specific to the particular causative factor.

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

So, for fledging database researchers, I wouldn't worry too much in the first year or two about having lots of publications and so on. It takes a while to get started. So, settle down, maybe write a grant proposal, get comfortable with teaching, find students, don't set high expectations about publishing two big papers a year during those first couple of years. Be easy on yourself as you ramp up.

By mid-career are you suggesting before or after tenure?

Either way.

Ok, so leading up to tenure, I think it's important to focus on the tenure process. One of the nice things about having tenure is that you can choose these arbitrary pet projects and even choose to spend most of your time on those and you have the academic license to do so. Before tenure, there is maybe a little bit of a risk if you spend a majority of the time on those because if they don't pan out, you would not have enough to show. So, maybe limit yourself to one pet project before tenure and maybe branch out afterward. Coming up for tenure, it's important to have your work known by the community. Give talks about your best work and visit other labs. The crucial thing about the tenure process, having seen it from both sides, is the quality of the letters of recommendation. You want to get letters from people who know about the impact of your work. So, tell people about the impact of your work at conferences. Do the circulating among the people in the field, particularly the senior people. Give them the elevator pitch if necessary or try to sit down with them for longer periods and communicate your work to get it as well-known as possible.

The crucial thing about the tenure process, having seen it from both sides, is the quality of the letters of recommendation. You want to get letters from people who know about the impact of your work.

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

I guess I have several favorites. I like the well-founded semantics I did in my Ph.D. thesis because it had high impact. Even now, it has many citations[8] and it sort of resolved a question that many people had posed for a while. So that was satisfying, and I enjoyed that work for that reason. Some other work that I like, I particularly like the cache conscious B+ tree work that I referred to earlier. I think we got in pretty early. I don't think many people at that time appreciated how important the cache was in the database community. I think we were trendsetters in that regard and this particular paper has influenced how people design indexes now and now it's regularly routine to make indexes cache sensitive or cache aware in various ways. Maybe it's too early to think of this biology paper that I published at PLoS One as my favorite, but this was sort of a major undertaking, it was a lot of fun doing and a lot of work, and it's something brand new. I have a fond feeling about it.

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

[8] The PODS paper has 591 citations, and the JACM paper has 2042 citations.

Okay, so I imagine many people would say they'd like to do more coding and that as a faculty member you would distribute the coding tasks to your students and not code yourself. I actually do find that I get time to code and I like coding and programming and so on. That is something I already do. I think the thing that I would like to do beyond that is to explore new domains. For example, in this biology application, I'm reliant on these other people doing the wet lab experiments and do the sequencing and so on and I know I have some colleagues at Columbia who started out as geneticists doing the theoretical work and basically evolved overtime and took courses to master the wet lab work and so on. I think it would be fun to do that sort of thing, to try to learn the technologies dealing with biological reagents and so on. I think that would take a fairly big investment of time and I'm not sure I have the time to do that, but if I had spare time, I think it would be fun to get to that point that I could be competent at doing those things and eventually direct others to do wet lab experiments in support of these biological hypotheses.

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

Okay, so there was the big biological revolution when the genome was sequenced where everyone was looking at these questions, sequencing and so on. At the time when that happened, I actually questioned whether I had chosen the right field. I thought to myself, okay if I had been doing my Ph.D. ten years later, might I have chosen to go into genetics or bioinformatics or something like that instead of computer science? In retrospect, I think I'm in a good position now because having done database work and being able to pick up a lot of this biology, I can make contributions that kind of span the technology and the science. A lot of the stuff that came out early in the genome revolution was technology that became obsolete over time. Things change a lot, and if you end up investing too much in a particular technology and with time that goes obsolete, then that is not so useful knowledge. That was hard to see at the beginning of that time. So, I think maybe if I were to change something, it would be to learn more biology sooner, to be able to work on these problems, but I think that at least in the subproblems that I've been working on, I've been able to catch up so to speak.

[…] having done database work and being able to pick up a lot of this biology, I can make contributions that kind of span the technology and the science.

Thank you very much for talking with me today.

You are welcome.