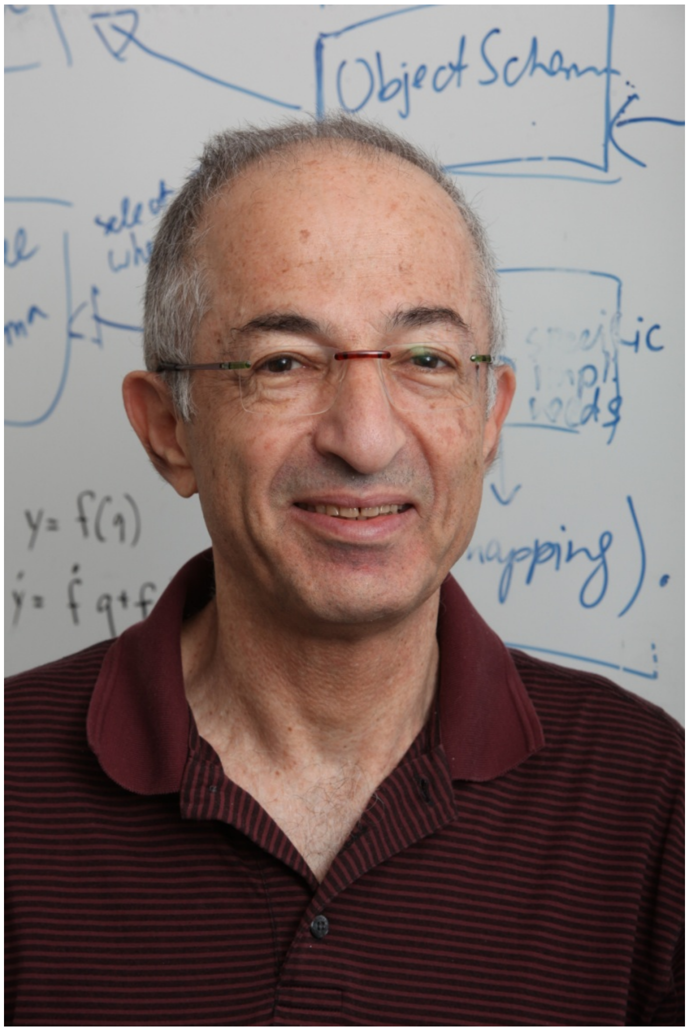
Catriel Beeri Speaks Out

On his favorite pieces of work and on the importance of Sabbaticals

by Marianne Winslett and Vanessa Braganholo



Catriel Beeri

http://www.cs.huji.ac.il/~beeri/

Welcome to ACM SIGMOD Record Series of Interviews with Distinguished Members of the Database Community. I am Marianne Winslett, and today we are in Providence, the site of the 2009 SIGMOD and PODS conference. I have with me Catriel Beeri, who is a professor of computer science at Hebrew University of Jerusalem. Catriel's research interests lie in database theory. He is a former member of the editorial board of ACM Transactions on Database Systems, and he is an ACM Fellow. Catriel's PhD is from the Hebrew University. So Catriel, welcome!

I understand that you are interested in programming languages. What do you think of our own community's programming language, SQL?

It's a nice language. I think it was nice from the beginning. Of course, now we have very thick manuals – the language has evolved in many different directions and I think that we… at least I don't find it so interesting now to read very thick manuals of SQL. But I think that the basic idea, the basic principle of SQL is the right one, and it is a very successful one. Obviously, this is the foundation for the success of database systems in the last few decades.

What do you think of this new push-for map-reduce-based programming?

I am all for it. You may know that I have been interested in functional programming languages specifically, not just in programming languages, for various reasons. One of them is because functional languages can actually express operations like map and reduce, which in some sense are more general than relational algebra, and they have very nice properties. They are completely declarative, they allow you to parallelize… So, the idea is not new... The idea of using map and reduce, or using functional languages as the basis for parallelization, it is not new, and maybe it has been waiting for the right time. And the right time is now, because we have multi-core processors, and programming languages like JAVA or C# are not going be up to the challenge of exploiting the parallelism of new hardware.

On the other hand, database people have been exploring the power of relational algebra -- not always in parallelization, of course. There have been some efforts on, you know, doing database work on many CPUs, but I think the main effort was optimization. But now, it is time to think about doing parallel optimization, I think… using parallelization for database that run on a single unit of hardware, because a single unit is going to contain many, many processors. So map-reduce and the relational algebra, I think, fit together and for me it is one paradigm. It is not that this is a new paradigm and SQL is an old paradigm. I have been looking at map-reduce at 20, 25 years ago.

I guess the map-reduce guys might argue with you and say that one of the differences SQL tries to hide from the programmer, what's efficient and what's easy to do, versus what's hard to do, like join ordering or something, and that map-reduce exposes what can be done efficiently and encourages people to program in a way that's efficient.

Maybe, but of course, once you have a nice paradigm, you can exploit the paradigm in different ways. You can try to embed the paradigm in different approaches to programming, so that they are good for different kinds of programmers or users. SQL was originally written with the intention to be used by people who don't know much about programming. It is still true that database programmers, or people who use databases, do not necessarily know much about programming. Sometimes they do, but they don't know much about query optimization, and they are not supposed to, because query optimizers are so much better than human optimizers. In this sense, I think the database story has been a success. Databases can optimize queries much better than humans, and this obviously will be true for map-reduce. Map-reduce is parallelizable. This is one part of the new buzz word. Map-reduce runs on many, many CPUs -- it is parallelizable. And again, it is not something that a programmer is supposed to do. It is done by the machine. The machine decides how to parallelize it.

What about impedance mismatches?

As far as I know, this has not been yet solved in a satisfactory manner. Originally, people thought that object-oriented databases would do the job, and of course, lots of work went into this new kind of new systems. And then they disappeared, almost disappeared from the marketplace. Relational systems are still there. There was a lot of talk about object-relational systems, but the basis for data storage is still relational systems. From what I see on the internet, from what I hear from my students, the impedance mismatch is still there. Work is still being done. For example, the new product LINQ from Microsoft seems to be very promising because it is embedded into the programming language. It is part of the programming language, and it has some features, which are very similar to SQL. So maybe it is not the ultimate solution, but it shows that people are still trying very hard to bridge the gap, and allow seamless programming. General programming on one side, which today is object oriented, and database programming, which is relational, and somehow the effort to combine the two is still ongoing. This solution, for example, does not come from the database community. It comes from the programming language community, and most of the talk on the internet (on this issue) seems to come from people with interest in programming language design, not from the database community.

I've heard from people in industry that a lot of people use those database programming environments. They start by extracting all the data into the programming language, and then they just compute on it, so they use the database like an object store, and they don't even issue any queries at all, which I find kind of amazing.

Well, yes, but of course it depends on the expertise of the people, and the size of the system. You cannot really expect all people to use database systems in the same way. In particular, the fact that you can express so many things in SQL, so many computational processes, so that you can delegate a lot to SQL, is not something that most programmers know. Programmers in large companies, they are normally either C# or JAVA programmers. They do not know much about database programming or SQL. There use SQL to store data, simple queries, but this is it.

At SIGMOD this year, we have the Celebration of 40 Years of the Relational Model, so I think it is a good time to look back at what we've accomplished. So, among the database theory results of the past 40 years, what would you single out as having had the most impact?

Well, that is difficult to say. I think one important result, important for the database theory community, it is not clear to me whether it has been really used in the general database community, but obviously, the idea of acyclicity in queries is important. It means that queries that cannot be efficiently computed, in general, can be efficiently evaluated if they are acyclic.

So this is the general idea, and then you know, there are additions and extension, and a lot of work. Query containment, and conjunctive query evaluation are very important for many other areas of database research in general, like data exchange, data integration, evaluating queries on views, etc. So, many, many different areas actually rely on being able to evaluate conjunctive queries. So knowing about the largest sub-class, which is efficiently computable, is a nice tractable computational problem, is important.

In a completely different direction, I think that transaction management is very important. The original paper is about two phase locking. I think this is a very nice result, very interesting, very nice, very important result, even though the paper originally appeared in, I think, the Communications of the ACM, if I remember correctly.

That's ok.

Certainly, not in a database theory forum. But there has been a lot of work in the general database community and in database theory about transaction management. I think this was one of the first, maybe the first paper that showed us that you can do interesting theory. Maybe not the first one, but the one that actually attracted a lot of attention, and obviously it had a lot of practical impact. OK, so if you are talking about important results, I think that acyclicity is important, for me, data dependencies and the chase are important.

OK, tell our younger readers about data dependencies and the chase.

Well, the story actually started when I was a visitor at Princeton, in '78. I worked on multivalued dependencies at the time, and I got Jeff Ullman involved also in the story. Then, we started to look on the subject of lossless decompositions. I think the chase came up in a very small way even in the paper we wrote at that time on lossless decompositions, but then Jeff involved other students, like Alberto Mendelzon, David Maier, Sagiv and Yannakakis. The first three wrote a paper about the chase[1], and I think this was the first time the chase formally appeared in print.

Then I went back to Jerusalem and I was very lonely, nobody to talk to: nobody in databases, and very few people in computer science. I was trying to continue the work in data dependencies, and database design. I did some work on design, some materialized, some not… The work on data dependencies slowly evolved into the idea that actually you can generalize it and talk about general dependencies. So I was one of the people that found out that actually you can first of all express the dependencies in first order logic, and then you can generalize and find out more general formulas which have nice properties. Essentially very closed in spirit to horn formulas and this is why they are so nice and have nice properties.

Then, when Moshe Vardi came along as a student, we formulated the chase (for this general class). Obviously, there were other groups that were working in a similar direction. Fagin was

[1] David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing Implications of Data Dependencies. ACM Transactions on Database Systems 4(4):455-469, 1979.

working on data dependencies in a very similar direction. But, for me, this was an important period. On one side, I was lonely, but on the other hand, I managed to come up with a nice idea and the chase, and the paper I authored with Moshe Vardi[2] at the time about the properties of the chase, I think was important. I remember it very fondly, and I think other people have found it as a very useful paper. It provides the fundamental properties of the chase in the general framework. I am very happy to see that now the chase and data dependencies are being used in other contexts. For example, they are being used by Val Tannen and his students for query optimization, and more recently for the work in IBM Almaden on data exchange, which seems to attract a lot of attention and I think it shows that data dependencies are, in some sense, a general idea that can be used in many different directions.

So that's the CLIO and the CLIOLLO project?

That's the Clio project, and I understand that IBM was planning to come up with a product Cliollo, for data exchange, which essentially is based on these ideas. So this is very satisfactory for me.

Among other potential major results of the past 40 years, did you want to say anything about schema design?

Well, schema design was actually my first work on database theory. Historically, I finished my PhD in Jerusalem. It was on automata theory, like almost everybody else at the time, because Jerusalem had the world experts on automata theory at the time, like Shamir and Rabin. So I worked on automata theory. Then I came to Toronto as a post-doc, and I shared the room with somebody named Phil Bernstein, who was a post-doc in Toronto. He stayed in Toronto for another year as a postdoc, we shared the room, and my first step was to read his thesis. So I was introduced to data dependencies, then we had some discussions about his thesis, about database normalization, and we worked further on the subject. A few months later, I was able to prove some NP completeness results about schema design, which at the time was quite new. I was not the first one. Sylvia Osborne, I think, wrote the first NP completeness results. But of course, NP completeness nowadays is mundane, but at that time, it was still exciting.

Then I moved to Princeton for the second year of my postdoc. I gave a database course, and I talked about database design, and dependencies, and Jeff Ullman got involved, and then Yannakakis, and Dave Maier, and Sagiv, and everybody got involved with data dependencies, so there was a bloom of research a lot of research. For me, this was a very difficult period, I must tell you, because, again, I had a family, I was a young postdoc… I didn't know what I was going to do next year, so personally I was under pressure, and I was not sure that data dependencies was the right subject to work on. And I wasn't sure if, you know, tomorrow I could come up with additional results -- typical problems of a young postdoc. But it turns out that yes, the area bloomed, and there was quite a flurry of activity, and PODS was started after I came back to

2 Catriel Beeri, Moshe Y. Vardi: A Proof Procedure for Data Dependencies. Journal of the ACM 31(4): 718-741, 1984

Jerusalem, by the people that essentially studied in Princeton at the time, and got also interested in database theory. Database theory at that time was dependencies, there was almost nothing else in database theory.

So it must have been hard, because it seems like back then all these people were in Toronto as a real hotbed of database activity, and to a lesser degree, Princeton, and then, from what you said, you went back to the Hebrew University and not only were there not any database people, there weren't very many computer scientists, and of course, we didn't have the web and the internet like we do now.

We did have email. Our department was the first one in Israel to have email connection, when I came back, we already had email in Jerusalem.

So you could collaborate.

So actually, I finished my paper with Jeff Ullman via email, I added some additional results, I wrote them down and I simply emailed them. This was done via email. So email was very useful, because otherwise, the paper would have been much shorter.

Did you want to say anything about datalog, what about all that effort on datalog, magic sets?

Maybe we should try to go chronologically, because we talked about my first visit to the States, which was in Toronto and Princeton, and then I came back, and of course, ok, I worked on dependencies. But, you know, people worked on other subjects, and I kept looking at what Phil Bernstein was doing, and he was working on something which I never knew about, which was transaction management. I didn't know what it was. And again, being in Jerusalem, I mean, there was almost no way I could start working on it. I read the papers, but if you are not at the center of action, and you don't talk to people, then you don't really know what are the interesting problems.

So, my first sabbatical was in Harvard, with Phil Bernstein. This was the time when I got introduced to transaction management, and I did some work with Phil, and Nathan Goodman about transaction management. And another one of my very favorite papers[3]… I actually wrote only 4 or 5 papers about transaction management, but the first one was the outcome of this year of work. It is the J. of ACM paper, a very long one, very difficult. I think that maybe not too many people have read the paper, but it's a fundamental one, in terms of providing the theory of how you prove correctness of, let's say, non-simple transaction systems. In other words, systems which have layers or which are complex, how can you go about proving their correctness? For me this was also very interesting, and in a completely different direction, completely different -- another one of my very favorite papers. I took a long time to get it done. It took a long time to get it accepted by J. of ACM, but I still think it is a very good piece of work.

[3] Catriel Beeri, Philip A. Bernstein, Nathan Goodman: A model for concurrency in nested transactions systems. Journal of the ACM 36(2): 230-269 (1989).

Then I went back to Jerusalem, and after a few more years, I went to another sabbatical to MCC, and this was about a year after the first paper on datalog by Jeff Ullman and Yehoshua Sagiv appeared, about magic sets. The word magic sets actually appeared first in their paper in '86. And I came to MCC, and that was also a very intense, and a very satisfying piece of work. The language was LDL: logic data language[4]. It contained sets, and negations, and of course, if you have sets and negations, you can express paradoxes. So the problem was how to come up with satisfactory semantics, and I think in the end, we managed to come up with a nice semantics based on stratification. So it's another nice piece of work, and with it some other nice papers in this period, it was the MCC team, and also Paris Kanellakis came down and we did some work together.

So it seems in this story, the way you are telling it, sabbaticals played a really important role.

For me, yes. Every time I went on a sabbatical, I was lucky enough to get into an environment where there were new problems, and people were working on systems, and theory, and this interaction in systems and theory, and a group of people working on a new subject was important. Sitting alone and giving lectures in a university without interacting with a lot of people is not a good way of finding new problems or new areas.

Yes, it seems a lot of people now file for a sabbatical but then they don't actually go anywhere, they say the kids have to stay in school, or whatever, and they don't actually leave their home institution. Does that matter now, now that we have such good internet?

I think of course the internet has changed a lot, I mean, first of all, it is much easier to get papers. At that time, if you really wanted to look at a paper, you needed either a personal contact, or to be on the mailing list of an institute, and even then, it took time before you got the paper. Even if you get the paper, say you read the paper, the paper is not the full interaction with people. There is a lot of difference between reading a paper and talking to people in the corridor or room, and then being involved in the process of asking questions and getting answers, and trying to find out what is it all about. It is completely different. So actually, when I went to Harvard, I went with my family. But when I went to MCC, I went alone. My wife and the kids stayed in Jerusalem. My wife came for a short visit for a couple of weeks, but most of the time I was alone. And since then, in all of my sabbaticals, I was alone. So instead of taking full year sabbaticals, I took 6 months, or 3 months. So sabbaticals, to me, were always important, because they gave me the opportunity to visit people, to see a different approach, new problems, talk to people, talk to students, and develop. I am not sure I could have developed the same if I had just stayed in Jerusalem and tried to read papers to see what is going on. I think sabbaticals are important.

[4] Catriel Beeri, Shamim A. Naqvi, Oded Shmueli, Shalom Tsur: Set Constructors in a Logic Database Language. Journal of Logic Programming, 10(3&4): 181-232 (1991).

Thinking about another aspect of technology, I understand you collect music, and that has changed a lot.

Yes. I am not sure I am a serious collector, but in all my sabbaticals, in all my visits to the United States, or when I come to conferences, until a couple of years ago, I always go out to find music stores and try to find records that I don't have, or sometimes records that I have recommendations for. Now it is of course CDs. It has always been CDs in the last 20 years. So I remember there was the PODS conference in San Diego a few years ago, and I took Tova Milo along and we went to a record store because I had the recommendation for a certain set of CDs of Haydn which I was looking for and I couldn't find in Israel.

Can't you just order everything from Amazon?

Nowadays, yes, but, Amazon didn't exist 10-15 years ago. Nowadays, everything has changed of course. I understand that now it is very difficult to find in New York a store that sells classical music. Somebody told me this yesterday. I came to New York, and all the record stores were closed because of Tower, now Tower is closing down[5], and you can't find real classical records anymore.

I didn't realize Tower was closing down.

I don't know if this is true or not, but obviously the world is changing, but you know, I remember that I used to go out in different small communities and towns and look for record stores, and you could find, you know, usually you don't find a lot of classical music in the suburban music stores, but you find a little bit. So my first Jascha Heifetz, I found in such a music store, in New Jersey. So I have a large collection. It is limited, I cannot afford to buy much more. I don't have place.

Space limited.

Yes.

So as soon as they change the technology to a smaller form factor, then you can expand again.

Then I can expand, yes.

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

Well, I think there are two advices. First of all, be open. Look out in different directions, and talk to people. Try to see what people are doing, and what is bothering people, and maybe you can find some model or answer or formulation that will make life easier and of course advance your research credit. I think I've always benefited from being involved with other people and trying to

[5] Tower used to be a retail music chain based in California. It is currently an international franchise and an online record store.

understand what was bothering them. In other words, I never thought that theory for theory is the right approach. Of course, there are people who do theory for theory who are very successful, but I thought theory for helping formalize what other people are doing is the interesting part.

Among all your past research, do you have a favorite piece of work? I guess you have mentioned several favorite pieces of work, but maybe there is one that we haven't already talked about.

I mentioned a few, but every time I think about it, there is another one. Of course, the paper about transaction management in J. ACM is one that I like very much. There were quite a few. I mean, there was a paper about magic sets with Raghu Ramakrishnan[6] that at the time was very nice, a very nice piece of work. There is another paper with Raghu that was never published, that I did when I visited him in Wisconsin. Because they wanted just a little more, and just a little more, and when I actually did this, well, the paper just never got published. So this is one direction. Then the LDL paper was a nice paper at the time. The LDL language was actually used by Dick Tsur and others for many years to come, and this paper about the language and the semantics was nice. I mentioned the dependencies, the chase the paper about the properties of the chase was a nice piece of work that I really liked, and it seems like some people actually like it even today. So there are quite a few.

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

That is difficult. I guess I would like to do a good piece of research in programming languages. Because I have always been reading programming language theory papers, but I somehow never got around to actually solve an interesting problem in programming languages. So I teach programming languages, and I read, but this one last step of doing actual research and providing a good solution to some interesting problem is something I never got around to do. So I would like to do that.

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

I guess I would have tried to be a much better programmer earlier.

Why?

Well, I am not sure I have a good reason. I always feel that my students know programming so well, so they can develop all sorts of interesting programming systems, and I cannot. So I do theory. Is it because I am good at theory, or because I cannot do implementations? It is not clear. I always think that maybe next year I will sit down and devote a few months to actually just writing programs, and see what happens. Then I can supervise students doing more interesting, maybe systems work. But I am not sure this is a good reason. I think altogether, I have done

[6] Catriel Beeri, Raghu Ramakrishnan: On the Power of Magic. PODS 1987: 269-284.

some good theory, and I know theoreticians who don't know how to program in even one language.

Well, we won't name their names here, they are to remain anonymous!

No! But they are very good, so it is nothing to be ashamed of. You cannot really be a professor and a good programmer, and I think this is the truth. You cannot be a researcher, and a theoretician, and a professor, and a teacher, and devote 5 hours a day to programming.

That's true, it is very time consuming.

It is time consuming. So there is a trade-off.

Well, thank you very much for talking with me today.

You're welcome.