Transcript of an interview with

Michael Stonebraker

2014 ACM Turing Award Recipient

Interviewed by Burton Grad,

Software History Center, Oral History Project,

August 23, 2007

BG: Burt Grad, Interviewer

MS: Michael Stonebraker, Turing Award Recipient

Abstract: Michael Stonebraker has been one of the most creative innovators and serial entrepreneurs in the relational database management field (and in other software areas). He was a primary player (along with Gene Wong) in the design and development of INGRES while at the University of California at Berkeley in the 1970s and in founding of the Ingres Corporation with Gene and Larry Rowe in the early 1980s. Ingres became a major player in the RDBMS marketplace competing against Oracle and the other significant companies in the field. In the early 1990s he was the leader in creating Postgres and Illustra which extended the use of the relational structure into areas other than business data. Then he was involved as an executive with ASK, Cohera, Vertica Systems and Streambase. He continues to be an active leader in a multi-university technology project in the New England region.

BG: My name is Burt Grad, and I'm interviewing Michael Stonebraker at his home in Moultonborough, New Hampshire. It's August 23, 2007, and this interview is part of the Software Industry Special Interest Group's Oral History Project for the Computer History Museum. Michael, I'd like to start with some family background, where you grew up and so forth. What were some of your interests while going to school?

Personal Background

MS: I grew up in Milton Mills, New Hampshire, which is about 50 miles from here, over on the Maine border. My parents moved to Massachusetts when I was ten. So the rest of my teenage years we were in suburban Boston. I went to Princeton University as an undergraduate. I'm exactly the age where when I got out of college, my choices were go to Vietnam, go to Canada, go to jail or go to graduate school.

BG: Let me go back. I don't want to move over this history so quickly. Family, parents, what kind of things did they do?

MS: My dad is an engineer. My mother was a school teacher.

BG: Where did your father work?

MS: Well, he worked for General Electric for a while and then my mother wanted to live in New Hampshire and so they moved to New Hampshire where, in the 1940s, there were essentially no engineering jobs. So he became a foreman in a fiberboard factory, until my mother passed away. That was the reason we moved to Massachusetts and my father joined Western Electric.

BG: So he went back to his engineering career at that point in time.

MS: Well, he obviously made more money as an engineer than as a foreman in a dying mill town business.

BG: You went to public schools in the Boston area? What suburb were you in?

MS: We lived in a town called Newbury, which is right next to Newburyport. And my father chose that town deliberately because at the time they did not have a high school and the town would pay the tuition for anyone who could get accepted at Governor Dummer Academy, which happens to be within the town boundaries. It's in the same general league as St. Mark's, Milton Academy and Browne and Nichols, those kinds of places. Both of my brothers and I got to go to Governor Dummer as day students with tuition paid by the town, which would not have been financially possible otherwise.

BG: That was certainly very shrewd on his part. What were the particular subjects you were interested in when you were going to school?

MS: Well my SATs sort of say it all. I made 800 on the math SAT and a 600 on the verbal SAT, so it was obvious what I was good at.

BG: Was engineering of interest to you? Did you do special projects and things while you were in high school?

MS: No.

BG: None of the Westinghouse Science things or any of those kinds of programs?

MS: This was in the 1950s. The level of activity available in the school system now is much better than 60 years ago.

BG: Were there particular science subjects that you liked the most?

MS: Well, the Governor Dummer was a traditional boarding school in the English tradition. So there really weren't any choices. Everybody took Latin, math, physics, English, chemistry. Everybody took essentially the same curriculum.

BG: Did you have any siblings?

MS: I have two brothers.

BG: And were they basically going through the same program that you went through?

MS: Yes.

BG: Were they older? Younger?

MS: I'm the middle of the three brothers.

BG: This proves all kinds of things about the older-younger syndromes and so forth. Sports? Any other activities while you were there?

MS: There were compulsory athletics at Governor Dummer, so I played soccer, and I played basketball, and I played golf.

BG: Strong interests or just things you did?

MS: Things I did.

BG: Choice of Princeton University-- why?

MS: Stupid move. Well, I mean the reasoning at the time was that I realized I was obviously good at math. Good at math, physics, chemistry. I could have done anything in the sciences. It was probably my father's influence to say, if you become an engineer you have better job prospects, which is absolutely true. And so I wanted an engineering school that was in a liberal arts school so that it wasn't just a bunch of geeks. I wanted to be far enough away from home so my parents couldn't come to visit, and I could go anyplace I wanted to. I was a good enough student, and Princeton fulfilled those characteristics.

BG: Having gone to a "private school academy," was that of assistance to you in getting in to Princeton?

MS: Of course. Governor Dummer was an all-male school. Princeton at the time was an all-male school. So my social life definitely suffered and I'm sure I would have been better off going to a coed school, I mean I could have gone to Stanford and didn't.

BG: Did you consider going that far away?

MS: No.

BG: Economically, were there any specific issues that would have determined where you went, because Princeton was a pretty expensive school at that point? It still is, obviously.

MS: Yes. It turned out that my grandmother had left a trust fund that basically paid college tuition for both my brothers and me. So it wasn't an issue.

BG: So that was kind of a free choice. You had a choice to make on that.

MS: I could have gone anywhere.

BG: Where did your father go to college?

MS: He went to Michigan State.

BG: Talk a little bit about your work at Princeton and what you did.

MS: I was an electrical engineer and I was a typical college student. We worked hard, caroused on weekends, went on long road trips in search of women.

BG: Were there fraternities there at Princeton?

MS: At Princeton there were no fraternities. There are what are called eating clubs, which are sort of local fraternities that essentially everyone joins, so I joined an eating club.

BG: Did that provide you with lifetime friendships or people that you continued to be in contact with?

MS: Yes.

BG: People in the field here or just totally unrelated to the field?

MS: Unrelated. I have friends I stay in touch with from Princeton, but no one is an engineer.

BG: Were you involved in sports or any other external activity?

MS: The eating clubs had intramural sports so I played intramural basketball and golf, but I wasn't ever good enough.

BG: You were tall enough for that time period.

MS: But I wasn't very well coordinated.

BG: Did you take any significant other course or were you taking computer courses during that period of time?

MS: This was at the beginning of the computer era so there were no computer courses in 1961. And so the University had just gotten an IBM 7094 or 7090, something like that. So in EE we started to do programming, but this was well before there were any computer science departments.

BG: Did you use FORTRAN, do you remember, at that time?

MS: Of course. That was what you programmed in. That was all there was.

BG: Did you have a particular interest in the computer side of it, or were you still looking to be an EE? You got the degree in it but were you looking to do that kind of work?

MS: It was pretty obvious to anybody who thought about it, even though in 1965 there were no computer classes at Princeton, but it was real clear that computers were going to take over essentially everything, as has happened. Computers are incredibly important in EE and all other engineering disciplines. So it was clear that the future was in computing but there were essentially no computer science departments. I graduated high school in 1961. I graduated from Princeton in

1965.

Post Grad Education

BG: When you graduated, did you go directly to graduate school? You said you had some choices there.

MS: Well, my choices were jail, the army, graduate school or leave the country. This was while the Vietnam War was in full swing. The Vietnam War was on the nightly news every night and it is real clear that there's a real analogy between the situation in Iraq right now and the situation in Vietnam at the time. I knew I didn't want to go to Vietnam, but the draft was still in effect and the lottery had not yet come into existence. So for any thinking person who could go to graduate school, that was a wise choice. And this was still in the days when the government was in their post-Sputnik era, we've got to catch up in the sciences. I had an NSF fellowship to attend graduate school so, basically, it was clear that I was going to sit out the Vietnam War in graduate school. So I went to grad school, to the University of Michigan in electrical engineering.

BG: Did you choose that because of its computer science work? Was there any relationship there?

MS: No, I chose them because they gave me the best fellowship. I could have gone to Stanford, Berkeley, MIT or University of Wisconsin.

BG: So your work at Princeton, you had done very well then?

MS: Yes. I went to Michigan and joined the CICE program, Computer Information and Control Engineering, which was sort of a joint program in engineering that was focused on computer science. There's also a computer science department in L&S at Michigan that had just been formed.

BG: Was Bernard Geller there?

MS: He was in the L&S Department. Bernie Geller was there. I quickly realized that anyone could call themselves a computer scientist at that time and so I just started taking computing courses and called myself a computer scientist. And you could still do that at that time.

BG: You did not get a Master’s degree. Did you go directly to a PhD?

MS: I have a Master's degree also.

BG: You do have a Master's degree. It didn't show up on your bio. So you got that when?

MS: 1966.

BG: That was in EE as well?

MS: That was in CICE.

BG: Were you teaching? Working? What was part of the fellowship program? Did you have that kind of responsibility then?

MS: It was a free ride, so I didn't have to do anything.

BG: Did you work with any of the professors in particular? And who was your thesis advisor?

MS: I was a TA because I wanted to be. I did a bunch of research projects. My thesis advisor was a guy named Arch Naylor. He's a control theory guy.

BG: And what was your thesis in? What was your PhD?

MS: My thesis had to do with Markov processes. You can think of it as mathematical operations research. I thought at the time, and I still think, that most PhD theses are a complete waste of time and I thought my thesis was a waste of time.

BG: That wasn't something that led to your follow-on work. That was really what I was driving at. Did that build toward something or lead towards something specific for you?

MS: I was in the situation where if I graduated and left graduate school then I would get drafted. So I had every incentive to stay in graduate school until the army didn't want me any more or until the War ended, whichever came first. And so, basically graduate school was not a time of great incentive to get out. So I graduated in 1971 when I was safely 26.

University of California at Berkeley

BG: Was that the key age at the time?

MS: Yes. I started looking for a job and this was right at the time when the NSF had just started a program called RANN, Research Applied to the National Needs. And the idea was that you ought to be able to use technology to site fire stations, help improve the court system, sort of public systems. Since I had done a thesis that looked like operations research and you could call yourself anything, I said well, I do that stuff. So Berkeley hired me basically to do public systems and the EECS Department at Berkeley was run by and continues to be run by very smart people who are very forward looking. They said, “This RANN stuff is probably going to go somewhere. We'd better get involved.” So I went to Berkeley in 1971 with the idea that I would work on public systems.

BG: Teaching was not a primary objective of yours at that point? But you didn't want to be a teacher per se; that was not your goal.

MS: Well, I think if you go into any name university, it's made crystal clear to you that you have five years to prove your value, to get famous and get tenure, or you get fired. And that was as true in 1971 as it is true today at all the name universities.

BG: Getting famous means producing, publishing?

MS: Yes.

BG: Rather than how good a teacher you are.

MS: Correct. So it was at Berkeley at the time, and continues to be true at Berkeley and essentially all other world class universities; you have to teach well enough not to be an embarrassment to the department, but there is no incentive to be an expert teacher. It doesn't help you get tenure.

BG: What were the particular courses that you were teaching in those first few years that you can recall?

MS: I was supposed to do public systems, so I started teaching a course on simulation. I taught introductory FORTRAN the first time it was offered. And I taught the course on operating systems. But I say that what was motivating me at the time was that within the first year it became crystal clear that public systems were really, really hard. So I did a project, it actually was NSF sponsored, to try to improve scheduling in the local court system in Berkeley. And it quickly became apparent that the data stunk and that getting good data was a huge amount of work. It was also apparent that the court system was basically a political system and there wasn't any way that it was going to run well because of the various factions. So it was pretty obvious that to get famous doing public systems were going to be really hard.

BG: What was the county that you were in?

MS: That was in Alameda County.

BG: Alameda County? So you saw the need for the data, both the input and the handling of the data in order to do anything, but there were these political overtones you saw in the way of being successful?

MS: Back in 1971, most of the records in the court system were not computerized, and so you collected them and computerized them and realized that there were so many errors that it was hard to use them to do better scheduling. The other thing was that it was clear that scheduling in the local court system was done so that you first scheduled to minimize wasted time for judges. Only when the judges didn't care, then you scheduled to minimize wasted time for the prosecutors and so forth. And so the thought that you would make 100 people sit there and wait so that a judge didn't have to waste any time. Why bother doing efficient scheduling when the system is scheduled perfectly for the benefit of one of the actors.

BG: So this was a project you started. Did you spend much of that full first year there on that or did you switch out of it fairly quickly?

MS: I also was interested in computer models of urban areas, predicting where growth should be. So I did a land-use model for Marin County, California. And that worked okay, but again, trying to predict where growth should go, there are all kinds of externalities that determine things that you have to model as special cases. Jay Forrester had done a model called Urban Dynamics, and it was clear that that style of modeling just wasn't going to work very well. And I also wrote a paper that took Jay Forrester's model, threw out half of it and still got the same results.

BG: Was that the dynamic programming work that he had done or what was it called?

MS: It was called Urban Dynamics.

First Database Work

MS: So Forrester wrote a book in 1969 or so that basically modeled an urban area, a specific aspect of it. So, I came up with, in the first year, two dry wells, and it was clear that my thesis work wasn't going to go anywhere. So if I was going to get famous I had to do something else. So I pretty much pitched doing public systems, pitched doing anything having to do with my thesis. Now you get a clean sheet of paper and five years to get famous. So I was looking around for something else to work on and Eugene Wong, who was at Berkeley at the time, said why don't you read Ted Codd's paper. And so I read Codd's paper and I read the CODASYL [Conference on Data Systems Languages] report, the first one in 1971. And the CODASYL report made no sense to me.

BG: This is the Bachman work you're speaking of?

MS: Yes. So I couldn't figure out why you would want to do anything that complicated and Ted's work was simple, easy to understand. So it was pretty obvious that the naysayers were already saying nobody who didn't have a PhD could understand Ted Codd's predicate calculus or his relational algebra. And even if you got past that hurdle, nobody could implement the stuff efficiently. And even if you got past that hurdle, you could never teach this stuff to COBOL programmers. So it was pretty obvious that the right thing to do was to build a relational database system with an accessible query language. So Gene [Wong] and I set out to do that in 1972. And you didn't have to be a rocket scientist to realize that this was an interesting research project.

BG: Were there others involved with you and Gene at that point or was that later on?

MS: It was just me and Gene.

BG: Had he been there before you or did he come in about the same time?

MS: No, he's ten years older than I am. So he was a full professor at the time.

BG: Had you connected with him before that? How did he happen to pick you is my question?

MS: No idea. Go interview him.

BG: We had hoped to interview him before, but haven’t been able to arrange it

MS: Because I think it is an interesting question, as to why did he decide to mentor an assistant professor?

BG: Was your mathematical work specifically unusual or unusually good?

MS: No. In fact, I don't consider myself a very good mathematician.

BG: Yet you had no problems in understanding what Ted Codd had done?

MS: Not to a computer scientist. The problem with Codd's work is that he basically has a first order predicate calculus with existential and universal quantifiers and anybody who doesn't have a mathematical background just looks at that and says, what the heck, this is Greek.

BG: But you have a reasonable mathematical background, so that's not a technical barrier?

MS: Perfectly accessible. No.

BG: You continue at Berkeley for many years, 25 years all together, going from assistant professorship from 1971 to 1976, to associate from 1976 to 1982, and then full professorship after that. Was the teaching part ever a significant portion of your interest or your work?

MS: No.

BG: You had to do it?

MS: You've clearly never been in academia, because you wouldn't ask that question

BG: I'm sorry. Let me make a point, Michael. It is not usual for us to have many people who are coming from an academic environment who have been very successful entrepreneurs. You're an exception to that. There are a few others but you're an exception. I'm trying to get that point across here.

MS: I think the situation is that in order to get tenure you've got to become famous. To become famous you've got to write a bunch of interesting papers. Those pretty much require graduate students to help you because you don't have enough throw-weight otherwise. To get graduate students to work with you, you've got to raise money. So you're basically running a little entrepreneurial research group and that's what every successful professor does. And getting tenure is all about making that process work and spending only as much time teaching as is necessary to avoid being an embarrassment. And once you get tenure, there are some people who decide to reorient their priorities completely, but most people continue in the same mode and the university highly encourages that because, for example, Berkeley as well as most other main universities doesn’t pay your phone bill. You've got to raise external money to pay your phone bill, have any secretarial support, and buy any computers. And so if you don't bring in research money, life is unbearable in most major universities.

BG: When did you raise your first research money on your own?

MS: 1971.

BG: You had done it right away?

MS: Well, yes. You have to.

BG: Was that NSF money at that point in time?

MS: Yes.

BG: Now how about the work with Gene Wong? Where did you get the research money from for that?

MS: The database money-- we had grants from NSF, we had grants from the Office of Naval Research, we had grants from the Army Research Office. We were hustling money.

BG: Was it a specifically for this kind of work?

MS: For a relational database, yes.

BG: So you had identified that as an area that might be of value to them.

Initial Development of INGRES

MS: Well in 1972, Gene and I decided to start building what turned into INGRES and we assembled a team of students to work on it and we immediately wrote grant proposals to support that work. And fairly quickly we had enough money. INGRES was written by one full-time programmer and three or four students between 1973 and 1976. We got enough research money to support that level of activity.

BG: Who did the design work on INGRES?

MS: Gene and I worked on it together. The architecture of the system was mostly me. Gene designed the query language QUEL.

BG: So QUEL was his?

MS: QUEL was his. And he designed the initial optimization strategies so the query optimizer was Gene's. I designed the execution engine and the file system: all the lower levels of the system along with Jerry Held, who's had an illustrious career in his own right. I guess he was at your meeting.

BG: He was at the meeting. He was a significant contributor, of course.

MS: He came on the scene fairly quickly, I think in 1974, and he and I did all the underpinnings of the system.

BG: You were doing this while you were still at the university. Did the university have ownership or feel it had an ownership of what you were doing?

MS: This is an interesting and raging debate to this day. I declared INGRES to be public domain software available to anybody and it was de facto public domain. What happened was that we got the system to work in the academic version, meaning we could get it to work and run a few queries by 1975. Most research projects declare success and move on to something else. You put

in the first 90 percent of the effort to get something to work and then you put in the second 90 percent to get it to really work. In 1975 and 1976, I don't know why, we made INGRES really work so that other people could actually get it to do something. By 1977, I think we had about 50 or 60 installations of INGRES around the world.

BG: Mostly at universities or were they at all kinds of places?

MS: Mostly at universities and research labs.

Academic Use of INGRES

BG: So they were using QUEL as their language at that point.

MS: Yes. That is what INGRES supported. So it was de facto public domain because we had sent it out widely. As to the intellectual property, what I have done forever is to just say everything I do is in the public domain and the university held no specific rights to the software.

BG: Was there a problem with them at the time?

MS: No.

BG: Has there been a problem subsequent to that?

MS: No.

BG: The University had no problem with that?

MS: No. The technology licensing office of every major university doesn't like that. And so at the time, all through the 1980s at Berkeley, the CS basically declared everything they did public domain from 4 BSD UNIX to Maxima. There was a bunch of really interesting computer science work, all of which was public domain. Universally the CS department just said what we do is public domain and the TLO didn't have enough political throw weight to get in our way. And academic freedom says that professors can decide what to do with the fruits of their labor, so the scholarly faculty stuff was all on our side, and so all the TLO could do was lobby to say, why don't you let us own it, and we would just say no.

BG: I was told in some of the other interviews and discussions that you were in contact with the people at IBM in San Jose through this period of time. How did that relationship develop? How did that work out?

MS: Well, Jim Gray, who we all know, knows everybody and well, it's hard to talk about Jim given his current status. But anyway, Jim is just a fabulous guy. Is or was, I don't know what we should say at this point.

BG: Just for the record, Jim Gray was lost at sea about six months or so ago, and there has never been any proof as to what happened.

MS: Jim Gray has a PhD from Berkeley and was around during 1971 and part of 1972, so Eugene and I got to know him and then he went to IBM Research and joined the System R team, and he's the kind of guy that just pokes his nose into everything. So it was mostly his doing that we would go to IBM Research in San Jose or they would come up here. So we probably met every six months, and so we knew what they were doing, they knew what we were doing.

BG: Now at the time you knew about the work they were doing in SQL.

MS: Yes.

BG: And Gene went ahead and created a different language, which he felt was better, I assume?

MS: Well, I'm sure you'll interview or have interviewed Don Chamberlin. Don came up with this language called SQUARE, in 1972, which is actually quite a nice, clean language but it had this problem that you couldn't type it on the terminals at the time because it required you to go up and down. And so then he designed a version of SQUARE that could be typed on a normal terminal, and that was the first version of SQL. From the very beginning both Gene and I thought SQL was a terrible language, and I'd say there's a very interesting article by Chris Date around 1985 that is an eighty page critique of SQL, and we understood all of that all along. I can go into why I think SQL sucks, but I'd say we thought we could do something way, way better. We knew about SQUARE before we built QUEL, but transmuting SQUARE into SQL and building QUEL were pretty much parallel activities.

BG: Were you seeing this that early as a business opportunity, or seeing this as a research type of thing?

MS: I was totally focused on getting tenure.

BG: So that was your objective and your goal?

MS: INGRES was my ticket to tenure, and that was the only thing of any significance.

BG: Gene was a tenured professor already at that point in time.

MS: That's right.

BG: Was his goal the research and the creation work?

MS: You should interview him because I would just be putting words in his mouth, and I think he's a control theory guy, very much an EE guy, and I think he viewed this as a way to broaden his interests, which he has very, very successfully done. But you should ask him that question.

BG: What I was trying to get at is whether that early you saw this as a business opportunity, an entrepreneurial opportunity.

MS: Absolutely not, absolutely not.

BG: OK.

MS: I can tell you how the entrepreneurial thing came about, but that was much later.

BG: Let's stay with that other thing. In building the system that you built, you were aware of the System R work.

MS: Yes.

BG: Were they similar models as far as the implementation was concerned, or were they just totally independent pieces of work?

UNIX and QUEL

MS: No, the System R team was much bigger. They had twelve PhDs, we had one; we had me and Gene, one fulltime programmer and some pickup students. They were forced to use VM/370, and they were forced to do a mainframe implementation. The data center at Berkeley had a CDC 6400, which was a batch machine, and it was obvious that was a non-starter to do a database system on, so we hustled enough money to buy our own computer, and what happened was Ken Thompson, of UNIX fame, turns out to have gone to Berkeley, and so through a series of lucky accidents we said, "Well, let's buy a mini computer and run UNIX on it." And so we got Ken Thompson to bring out UNIX and install it, and I think we were probably the first installation of UNIX outside of Bell Labs. That may not be true, but we were a very early one.

BG: What minicomputer did you get?

MS: We got a PDP 11/40. It was all we could afford.

BG: And that may have been a very good thing to implement INGRES on a much smaller machine.

MS: We were fundamentally focused on a minicomputer, on UNIX, we were coding in C, so we were a very early UNIX system developer. Basically we bought the UNIX way of doing things completely. We had a public domain system which we could send to anybody. The System R guys couldn't, had to get lawyers involved and sign non-disclosures to give System R to anybody. I think the implementations are very different, and I think collectively the System R guys focused on certain things, we focused on certain things, and largely the two implementations were complementary.

BG: Was there a discussion between the two of you at that time, of the SQL versus the QUEL languages? Was any attempt made to maybe bring them together, or integrate them in some way?

MS: Oh, they're not integratable.

BG: I see, just the nature of them is such that you couldn't have joined them.

MS: In my opinion, the big error that the SQL guys made was that SQUARE had this nice feature that you said: "select name from employee where department equals." So that's a block, and how do

you find the department, well, it's the department, “select department from department table where floor equals 1.” It's very nice block structured, so you had inner blocks and outer blocks, and if the inner block needed some more information you'd have an inner, inner block and so forth, so it was a very nice block structured language. The trouble is, and that was a lot of the philosophy of SQUARE, that first of all aggregates don’t work very well in this model, and secondly, it turns out that there are a bunch of queries that cannot be solved in this purely block structured fashion, and the whole idea was you evaluate the inner, and that produces an answer; you substitute that answer into the next inner and you go from inside to out to solve the query. However, there are a whole bunch of queries where that's not a general enough query paradigm.

So, what ended up having to happen was that SQL had to get extended, and I think SQL2 was in 1976, and so it put in the notion of in SQL1 you did joins from inner blocks to outer blocks, in SQL2 you could have multi table blocks, so that you could say “select yada-di-ya from employee,E, department,D” and you could have single block multi table queries. And that basically rendered the block structured notation in SQL1 obsolete, and instead of redesigning the language completely to get rid of the nested nature of SQL, they just tacked on the extra stuff. And so I think SQL could've used a total redesign, from SQL1 to SQL2 and they didn't do it, and you have to talk to Don Chamberlin about what he was thinking about at the time. Anyway, QUEL never had a nested structure. We didn't believe in nested structures, so the two languages were very semantically different. You couldn't have put them together.

BG: And Gene Wong was the one who conceptualizes the language then?

MS: Yes.

BG: Did you find areas or types of queries in which QUEL was not an effective way to work?

MS: In the early 1980’s when Oracle had commercialized SQL and Ingres Corporation had commercialized INGRES, a big selling feature of INGRES was that QUEL was a much nicer language than SQL, and I think that was generally agreed to by practically everybody at the time.

BG: Nicer in the sense of being able to form the queries more readily?

MS: Yes.

BG: Or better in the sense of leading to better search techniques?

MS: Well, both.

BG: So it was in both areas you feel it was advantageous?

MS: Yes.

BG: And incidentally, comments that were made at the RDBMS pioneer meeting, almost everyone there seemed to feel that QUEL was a better language; even Don Chamberlin may agree with that, I don't know.

MS: I don’t want to get too technical or we'll spend the whole afternoon on SQL, but if you have a block structured language it turns out that the natural way to solve it is from inside-out, as I said a few minutes ago, but that may not be the best strategy. In order to look at all strategies you’ve got to flatten the query into a single block, and then have an optimizer look at all the possibilities, which was what QUEL did. And so you want a flat language for the benefit of the optimizer, and the System R guys in fact eventually flattened all their nested queries, thereby rendering the whole nesting structure as completely unnecessary.

BG: And yet that was the underlying architecture that was built in, some things that they couldn't get around readily.

MS: Right. So I think that was one of, I'd say, the advantages of QUEL. It had a cleaner notion of aggregates. One of the things that mystifies me is that in SQL, if you have an outer block and an inner block, so you can say “Select name from employee where department equals, select department name from department where floor equals 1,” so remember, there's an equals between the two blocks. Take the same query, take out the equals and put in the word “in,” “where department in” and then there's the inner block. Well, you would think under any ordinary circumstances that those two queries would have the same meaning, but they don’t. And so for whatever reason Don built in some very strange semantics into SQL that annoy programmers to this very day.

BG: I'm going to push ahead, as you suggest. Okay, so we're now in 1976, you have a reasonably good version of INGRES working. You have fifty, sixty locations that are using it, pretty much on PDP11’s or equivalent machines, or anything that ran UNIX.

MS: Well, the only things that ran UNIX at the time were PDP11’s. So this is all on 1140’s or 1145’s and 1170’s

BG: Were you providing any assistance or consulting help to these people who were using this system?

Starting Ingres as a Company

MS: By about 1977 we actually had a pretty significant commercial user, which was New York Telephone Company. And there was a guy named Dan Gielan, and he picked up the INGRES code and actually used it in a real world project. And we then realized what support meant, because there was a big difference between people who just played around in academia and people who actually tried to get useful work done. So basically, with our one fulltime programmer we were providing support in some sense, as well as doing more development. But the real turning point came in 1978, when Arizona State University considered INGRES for a student record system for the university, all forty thousand students worth. And they could get over the fact that you had to go get an unsupported operating system from AT&T in North Carolina, and they could get over the fact that you had to get an unsupported database system from these goofy professors in Berkeley, but the project went down the drain when they realized that there was no COBOL available for UNIX, and they were a COBOL shop. So we were always asked” “Who's your most serious user of INGRES?” and we'd be forced to admit hardly anybody because of an unsupported operating system, an unsupported database system, and no COBOL.

So it was clear that if INGRES was going to go anywhere we had to move it to an operating system that was supported by a real company. In 1979, Larry Rowe had joined the INGRES project, and we started to try to figure out how to commercialize the product, and that was in 1978, 1979.

BG: Where did Larry come from?

MS: He was on the faculty at Berkeley.

BG: So he again had been a Berkeley professor?

MS: He arrived, I think, in 1976 or thereabouts.

BG: Is he of a younger generation?

MS: He's about five years younger than I am.

BG: He becomes a key part of the team at that point in time?

MS: He becomes the third professor on the team.

BG: How do you proceed, what did you do then?

MS: Of course I had no idea how to start a company; I had somehow been put in touch with a guy named Jon Nackerud, and he was at that point the western regional sales manager for Cullinet, and he was selling IDMS. I went to see him and said, “How do you start a company?” He immediately was willing to join the team, and he had experience hustling venture capital money, so it was largely his tutelage that got Ingres Corporation off the ground.

BG: So Jon and the other three of you, the four of you were the founders then?

MS: Yes.

BG: And you originally called it Relational Technology Inc.?

MS: Yes.

BG: But the product name was INGRES from the beginning?

MS: Right.

BG: Where did the name come from?

MS: Well, you can't, in 1972, start a research project in the university without having a name, and so the name came about in 1972, and Gene Wong had some name consultant who thought it up, and it was INteractive GRaphics and REtrieval System--INGRES.

BG: I see. Which came first, the name or the acronym?

MS: You have to ask Gene. That came from him.

BG: Okay. Now, at this point with the university, because you're going to start a company or form a corporation that's going to use this thing, is there any problem with that? And was there no problem that the three professors will be forming a corporation and working for that part-time?

MS: Berkeley has a policy that faculty is welcome to consult on the outside up to one day a week, and we were simply doing it for our company as opposed to other companies. So no, the university had no problem with this.

BG: That was not an issue then?

MS: No.

BG: So when was Relational Technology Inc. formed?

MS: It was in 1980.

BG: You all formed it as a stock corporation. Did you have outside money? How did you do it?

MS: We had venture capital backing from some folks called Sutter Hill Ventures. We had pitched to them and a bunch of other VC's and so they backed the company.

BG: Tell me about this experience, of getting this stuff ready for the VC's and creating a new business, that was a new experience for you.

MS: Totally new.

BG: How did you feel about it, what role did you play?

MS: Well, it was clear immediately that of the three of us I was the best technical salesman. A lot of it was like the VC's say, "Well, if this is such a good idea why won't IBM release theirs next week?" So you have to explain to them that software is hard to build. So the VC's by and large were not familiar with software, they had mostly done hardware startups and so you ended up having to teach them why software was hard to build.

BG: Well, you're at a point in time when Cullinet (which had gone public in 1978) was the first software company that was listed on the New York Stock Exchange in 1982.

MS: That's right. That was around the same time.

BG: And that was a real breakthrough because I was very active in ADAPSO [Association of Data Processing Services Organizations] and we just couldn't even get bank money at that point in time. So you must've done a heck of a selling job to get the VC's to put money in at that point.

MS: I think another way of looking at it was Sutter Hill Ventures got a really good deal, so I think their point of view was almost certainly like squeeze them hard, don't put in very much money and so if it goes belly-up so what, and if it works we make an unbelievable return on our money, which was of course what happened.

BG: Can you tell us how the percentage was split between Sutter Hill and the four of you?

MS: I could reconstruct that, but the market cap of the initial round valued the company post money at less than a million dollars. This was 1980, and so I think by today's standards the deal stunk for the entrepreneurs.

BG: Did they get half the company? Were they in control or were you people in control?

MS: I've now done five startups, and the VC's essentially never agree not to be in control, at the end of the day.

BG: So regardless of what these stock shares were, they would be controlling the direction?

MS: I think in the case of startups, at the end of the day that's the only thing the VC's can do. The VC's had very, very little power over any company I've been involved in, because you just tell them no and what are they going to do, fire me. So I've always been un-fireable in any of these ventures, meaning that if I said I wasn't going to do something and they said "Yes you are," then I would quit, then they have nothing. So, I've always held a special position that was very different than in the typical company. I think the only time they have power is when the company runs out of money, and then you have to raise money and then they have power, but I find to a first approximation their power is pretty limited, in any venture I've been involved in.

BG: What was your title then when you formed Relational Technology?

MS: CTO.

BG: You were CTO. Who was the CEO at that point, do you remember?

MS: It was Jon Nackerud.

BG: Did the others have formal positions, Larry and Gene in the company?

MS: Yes, we were all technically consultants one day a week, doing various things.

Grad: Did Jon bring in marketing people then or was that too early?

MS: In 1980 we started operation in Jon Nackerud's basement to save money. The whole idea was to port the INGRES code to VMS, because UNIX was unsupported, there was no report writer and no utilities and no serious documentation. We had to port the system and then work on getting the system to go faster. We spent at least a year in heads down engineering, moving the system to something that we could sell. The company was Jon Nackerud plus a bunch of engineers for the first eighteen months, and then we started to add people.

BG: And you had enough money from the initial funding to carry you that long?

MS: First of all, we lived in Nackerud's basement, so this wasn't a garage operation, it was a basement operation. We didn't buy a computer; we used the one at Lawrence Berkeley Labs. We lived lean beyond belief.

BG: But you had to pay the programmers?

MS: Yes, but we only had four or five.

BG: A very small team.

MS: That was all we could afford. We made the $350,000 investment last, I'm going to guess, two years. Gene would probably know better, but this was a shoestring operation.

BG: You say you were working a day a week each of you. What were you actually putting in?

MS: Probably a day a week.

BG: That all? It didn't become dominant as far as taking over your life?

MS: Early on we hired a guy named Paul Butterworth who came to your pioneer meeting, and he was the chief programmer. Really the three of us did technical marketing and watched over the engineering team, and you can do that in a day a week.

BG: You kept the name Relational Technology for a while. When did you change it, do you remember?

MS: 1983, some time around there.

BG: Because Oracle's original name was something very close to that, if I remember, like Relational Technology Systems, or something like that.

MS: Well, when we started to sell the product seriously, Nackerud said "Look, I'm the guy from Ingres, I'm not the guy from Relational Technology, so there's no upside to having a different name for the company than the product," and Oracle came to the same conclusion.

Early Competitors

BG: Now Oracle had been in existence a little earlier than you, 1977 or 1978 sticks in my mind as being about the right time when Larry Ellison formed the company.

MS: They were slightly ahead of us, yes.

BG: Were you aware of them and what they were doing when you formed your company?

MS: Oh sure.

BG: Was that a factor to you? Was that a consideration of any kind, at that time?

MS: Well, Larry was out in the marketplace when Oracle didn't work, saying he was ten times faster than INGRES, which did work. And so that kind of got my shackles up. So the interesting thing about Oracle and Ingres was that Ingres revenues, in 1982, 1983 and 1984, went one million, four million, and nine million dollars. We did nine million dollars in 1984. I think that's right, Gene Wong would probably know better. But even though Oracle had been in the market earlier we were gaining on them substantially, so I think Oracle did eleven or twelve million dollars in 1984. So we were gaining on them and we had a better product, I mean practically everyone agrees that the Oracle product just stunk at the time.

And so we were gaining on them, and the key event which I think is a complete watershed, was in 1984. IBM announced DB2, and up until then they sold SQL/DS on DOS, and that was considered a non-engine to anybody serious. And so they had a dual database strategy which was to run IMS on MVS and run whatever you wanted on the other operating systems. And so in 1984 they announced that DB2 was going to be available on MVS. And at the time IBM had way more throw weight than they have now, and so in one "swell foop" they had enough throw weight that that meant SQL was the answer, and anybody with any other query language had to convert to SQL. Larry Ellison very skillfully marketed the heck out of: "We're selling SQL now," and even though Ingres got SQL running quicker than anybody else, in 1985 Oracle's revenue skyrocketed and Ingres's didn't. That was pretty much the end. Then Oracle was between two and three times bigger than Ingres, and they marketed that: "We're safe, we're the big guy, we're the safe bet," and they started employing what I call the America Cup strategy, which is whatever Ingres did Oracle would announce "Well, we're doing that too," even though they rarely did. It's sort of the standard Microsoft tactic these days. Oracle and Bill Gates are experts at selling futures to stall the market. Oracle did that very successfully. In my opinion, if IBM hadn't announced DB2, Oracle and Ingres would've switched places within a couple of years.

BG: Did you feel you had enough infrastructure in your company at that point to have really grown? That's a very fast growth rate that you were describing.

MS: Sure, sure. I say we were. IBM had to know the problems with SQL at the time, and they chose not to fix it.

BG: That was going to be my other question. They did not make the major changes you feel would've been necessary?

MS: I think they simply took the System R front-end and glued it onto VSAM and they didn't change the upper levels of System R at all. If Eagle had succeeded, or if they hadn't just taken the cheap and dirty route and glue the top half of System R onto VSAM, either of those two events would've caused just major implications downstream.

BG: I'd like you to talk a little bit about Eagle if you would. What do you know about it and so forth?

MS: Sure. If you're IBM and you're looking for an MVS database system and you've got one called

IMS, which is kind of a low level thing, and here's this high level interface, the obvious thing to try and do is stick the high level interface onto the low level system, which is what Eagle was attempting to do. And the trouble is that it technically didn't work, because of the design of IMS logical databases. One of the problems with IMS is that again in 1968, which was when it first came out, it was a very clean hierarchical system, and so their interface language DL1, was a perfectly reasonable, hierarchical, sort of low level query language. If they just stayed there they could've put System R right on top of that system, but what happened was that IMS customers immediately started whining that you have a hierarchical system and we have network data, we have CODASYL style data.

So IMS tried to react to the requirements for networks by creating these things called logical databases, which meant that underneath the sheets it was really a network, but DL1, which was a hierarchical language, still worked. And they did this with this kluge called Logical Databases. And that sort of screwed up the physical structure down in the guts that made it impossible to put a clean semantic query language on top of IMS. And it had to do with they couldn't make inserts and deletes work right, and so it just technically didn't work. And the moral to the story is that if you have a good paradigm, don't keep overloading it. Don't try to make a go-kart into a Ferrari, because downstream that's probably going to be a bad idea. They put this networking support into IMS and that made Eagle impossible, at the end of the day. The outcome from IBM's point of view would've been vastly better, if they'd been able to say "Look, we have this very high performance infrastructure, and if you want a go blindingly fast code against it. This is called IMS. If you want this clean, nice interface, you use System R, call it DB2, it runs against the same data;" it would've been a much smarter strategy if they could've technically made it work.

BG: I'd like to come back to one thing, just before we move ahead.

MS: Before we move off, I do want to make a couple of other comments about Larry Ellison.

BG: These are for permanent record; you do understand that, okay?

MS: Number one: he and Bill Gates share a lot of common features. The first one was that Larry Ellison had no qualms about lying to his customers. For example, at one point Ingres came out with Integrity Constraints, so they were in the product, and the Oracle guys said "Well, we have them too." If you looked at their documentation there would be all this syntax for integrity constraints, and then there would be a little note that said "Not implemented yet." So he had no qualms about mixing future tense and present tense, a characteristic he shares with Bill Gates. Number two, he invented the concept which I call "scorched earth," which is if his sales guys were going to lose a deal then he would cut the Oracle price to zero, and thereby render the winner to get essentially no revenue. So his sales tactics I think are somewhere between predatory and unethical. I think he skillfully and shrewdly capitalized on the DB2 announcement. But I think both he and Bill Gates have given the entire software industry a very bad name, because I say, the other thing about Oracle is that essentially to this day, and it's gotten better in recent years, they QA their product very poorly, and so no serious person will touch Oracle for at least a year after it initially comes out because the early customers do the Oracle QA. So, both companies release buggy products that I would not feel comfortable putting into the marketplace.

I guess another thing is that Ingres had a real query optimizer, System R style; it was about 1990

before Oracle had one. Oracle evaluated queries from left to right, and you had to write your queries in the correct order that would make them work well; so depending on how you wrote the query you could get anywhere from good performance to terrible performance. And one of the fundamental tenets of relational databases was you weren't supposed to have to do that, and yet you had to do it with Oracle. So Larry shrewdly marketed that as well: Ingres merely had a semantic optimizer; we have a syntactic optimizer. They were skilful at fud [fear, uncertainty, doubt]. But to this day the enterprise software market is dominated by those two companies. They both release what I consider crummy products routinely; they both attempt to stall the market by talking about future releases long before they exist and when they're not even committed to actually existing; and they give all customers reasons to not trust any vendor. I think their legacy on the software industry is going to be quite negative for a long period of time.

BG: Appreciate your thoughts. I'd like to go back a little bit on Relational Technology and forming it.

MS: Sure.

Financial Support

BG: Was there a particular partner at Sutter that had supported you or was your primary partner you worked with?

MS: First it was Len Baker, and after that it was Bill Younger.

BG: Were there second levels or third levels of VC funding for the company?

MS: The second round was led by Welsh, Carson, Anderson, and we got on the board a guy from Welsh, Carson, Anderson whose name I will try and remember. And then there was a C round and New Enterprise Associates was in that.

BG: Now, over what period of time are these three rounds taking place, the first one you said was 1980?

MS: The A round was in 1980, the B round was probably 1982, and I don’t remember when the C round was.

BG: When did the company go public?

MS: 1987.

BG: Okay, so those three rounds carried you until you went public?

MS: Yes.

Oracle

BG: At the meeting, one of the things that was said very clearly by the people who had been at Oracle, including Mike Humphries for example, is that Ellison’s strategy was to target specific companies

each year in terms of the one he felt was the greatest threat, and then to do specific actions that he felt would not just win, but would harm the other company, that was what was stated at the meeting. Did you feel that way as well?

MS: I think in the mid 1980s that company was Ingres.

BG: So that you were the one they targeted?

MS: Oh absolutely, and to a first approximation in the VMS market, we were the only two serious competitors. Informix played at the low end, but at the high end we were basically the only two.

BG: Sybase was not involved?

MS: They didn't exist yet.

BG: Anything more you want to say about what happens with Ingres after IBM comes out with DB2, Oracle starts to grow much faster. Did your growth curve sort of flatten out at that point in time?

MS: Well, in 1985 we went from nine million to sixteen million dollars, so the growth curve flattened out, and then it resumed about what it had been. But Oracle's growth curve accelerated and ours decelerated.

BG: Because nine to sixteen is still a pretty significant growth factor, but you really felt that the spread then changed the ballgame.

MS: Well, I'd say so. By 1987 Oracle was three times our size, and it was the differential growth rate in those couple of years.

Growth of Ingres

BG: And you went public in 1987. Were you making money at that point in time, was the company profitable?

MS: Yes. Back then you had to be profitable to go public.

BG: Yes. As against the late 1990s.

MS: Yes.

BG: So again, is that the sort of thing that Gene [Wong] would have more details on? Was he the financial person?

MS: He would probably.

BG: You were continuing your CTO role then during this period of time?

MS: Yes.

BG: Were there any other principal new players that had come into Ingres by that point that we should mention here? Jon Nackerud, was he still there at that point, or he'd left by then?

MS: Well, what happened was at the very beginning Sutter Hill said: “Nackerud is a good sales guy but he's not an administrator. You guys are obviously not administrators; you basically want a VP of operations.” And so they said “You guys should hire Gary Morgenthaler, who is now a VC. You should hire him.” He was basically managing the business, Nackerud was doing sales, and the three of us were watching over engineering.

BG: You say watching. Now as the CTO, you weren't the VP of development though.

MS: No, to a first approximation, that was Paul Butterworth, and much later we actually hired a VP of engineering, but that was much later.

BG: Other significant people at that point in time that you can remember?

MS: What happened was it became clear fairly quickly that Nackerud was not the long-term president, and so I would get gentle nudging from the VC’s, and I would just say “No, Nackerud stays.” Eventually it became clear even to me, and then Nackerud left the company. We did a search and concluded that we couldn't hire anybody better than Gary [Morgenthaler], so he got promoted to be CEO.

BG: How long did Gary stay?

MS: Until 1987 I think.

BG: Until the time of the public offering?

MS: About the time of the public offering.

BG: And what happened after that?

MS: What happened was, things were going okay but they weren't going great, and part of that was probably due to competing with Oracle. And so at some point we decided that we should hire a COO under Gary, and we did so, and that guy’s name was Paul something or other, and I can't remember his last name.

BG: I'll check it out.

MS: Gary was CEO and we had a COO, and that was actually I think what the go public configuration was.

BG: And you and Gene and Larry were all continuing to still be involved?

MS: Yes.

BG: What were some of the major things you were concerned about during that period of time from a technical standpoint? During the mid to late 1980s?

MS: How to compete against Oracle.

BG: Was that from a feature standpoint, or did you look at it from a marketing standpoint; how were you viewing it?

MS: From a feature standpoint. What could we do in the product that would give us a competitive advantage against Oracle. But as I said, it's very difficult when you're competing against an America Cup strategy, because whatever we did they would say, “We're about to have it too.”

BG: And that was one of the things that was said specifically at the meeting that we were at, that was in fact the strategy, when you announced something they would say “Yes, we have it also,” or “We will have it shortly,” whatever.

MS: Yes, so we put some very innovative stuff into INGRES.

BG: Give me an example of two or three of the things. Can you remember?

MS: We introduced a distributed database system, called INGRES Star, which would integrate data in multiple INGRES installations that you could scale out, and Oracle immediately announced “We have it too,” even though they didn't really. They were able to “fuddify” the market. We introduced the notion of abstract data types. We did very innovative technical things.

BG: I'm just trying to get a record of some of those things that you felt were so significant at that point.

MS: The biggest mistake we made, I think, was in response to Gary Kelley, who was then at Sequent, when he said, “What you guys want to do is do intra-query parallelism, so that we can spread a single query over multiple Sequent CPU’s.” So we had a little skunk works to try and do that, but it was never very serious. So Gary Kelly eventually got disgusted and after he left Sequent he went to Informix who did exactly the same thing. I believe that intra-query parallelism was largely why Informix was so successful in the early 1990s. That was an opportunity we screwed up. Ingres was trying to do too many things, and whatever we did Oracle would apply the America Cup strategy.

BG: But did they eventually implement those things even though they were later than Ingres?

MS: Yes.

BG: So once they saw the spec of what you were doing, they were able to imitate or copy that?

MS: Well, they would do crummy installation implementations of parts of stuff and say they'd have it next month when in fact it would come out in three years. I mean it was standard Microsoft-Oracle fud.

BG: Well, there are a lot of other companies who had that same approach before that.

MS: Yes.

BG: It was not invented by either of those two companies.

MS: That's true.

BG: We have a long record in the mainframe world. There was a company called IBM who was occasionally accused of doing some of those kinds of things.

MS: Yes. But I say, I think everyone will agree that throughout the 1980s INGRES was a universally better product than Oracle, and much more innovative in terms of bringing features into the database.

BG: Now there comes a point around 1991 where Oracle was in very deep financial trouble, and basically was very close to going out of business, we were told. Were you aware of that at the time?

MS: Yes.

BG: Did you find any way to take advantage of that?

MS: It didn't help Ingres' sales.

BG: Why not?

MS: Oracle hid it very well. We knew it, but if we told customers Oracle is about to go under they would say "Says who?" and so you say "Well, that's what the tom-toms are telling us," so.

BG: So you weren't able to take advantage of their weakness at that point in time?

MS: No.

BG: Then they brought in new management which apparently did clean things up from a financial standpoint, at least that's what we were told.

MS: Well, they also were close to going under I think in 1982, or 1983, because their product basically didn't work, and they managed to pull the rabbit out of the hat then.

BG: But that was before you were a significant company.

MS: Yes.

BG: Informix becomes a much more significant player in the late 1980s or early 1990s.

Sale of Ingres to Ask

MS: Well, I think so. In 1989 or 1990 it was real clear that Ingres had lost to Oracle, and there wasn't a whole lot you could do about it. So you had a viable company, but one that wasn't really going anywhere. The company decided to sell itself to Ask to basically partner with a big application of INGRES, and to get Sandy Kurtzig as president, who would be able to counter some of Ellison's charisma in the marketplace. Basically, we got bought by Ask, but to a first approximation very quickly it was really a reverse merger, because the Ask manufacturing application didn't really work, and so over the next few quarters INGRES became the dominant piece of the composite company.

BG: Did they buy using stock or using cash? How was that acquisition made?

MS: It was cash.

BG: It was a cash deal. Did you all stay involved in the company after it was bought by Ask?

MS: All I can do is answer personally. Sandy Kurtzig arrived on the scene. Her reaction to thin margins was to cut engineering, and the only reason INGRES sold at all was that it was a technically better product. So she cut engineering, which risked how long it could stay a better product. She also demonstrated that she had no solution to the marketing problem against Oracle. So that made the handwriting on the wall very clear that the company was not going to do well; so that's when I left. And it was eventually sold to CA, [Computer Associates] but that was after I had left.

BG: You stayed until 1992 with Ingres. Is that the year it was sold to Ask or is that afterward?

MS: It was sold to Ask before 1992, and it was sold to CA after 1992.

BG: Meanwhile, during these years, you're continuing your academic progress, becoming an Associate Professor and then a full professor at Berkeley. What other activities were you doing at Berkeley during that time?

Development of Postgres

MS: Ingres started in probably 1981 or 1982. There were probably forty research papers put out by the research community, all of which had the following theme: relational databases are supposed to be terrific, so we tried out using one in application X, and X is a variable with forty values, and guess what, it didn't work all that well because of Y, and there are forty Y's, and therefore here's the stuff we invented to fix the problem. And so there'd be a whole bunch of suggestions for enhancements

BG: Were these in particular industries or applications or some combination?

MS: These were CAD [Computer Aided Design], text processing, dot, dot, dot. So it became pretty clear that relational databases didn't work outside of business data processing at all, because these examples were all stuff other than business data processing. One of the places where they didn't really work at all was geographic information systems, and one of the main people at Berkeley

who was interested in Ingres was a guy named Pravin Varaiya who was in urban systems and was very interested in geographic data, and it just didn't work in relational databases. So I looked at these papers and saw solutions to various vertical markets, so I said “There ought to be a better way to fix relational databases than what people are proposing.” So, we came up with the notion of abstract data types, which are now in all the relational database systems, and we wrote a paper in 1983 that sort of said here's how to do a much more general way of extendibility in relational databases. Up until then we had been extending INGRES, so we still had the university version of INGRES, and we still had a fulltime programmer and three or four students, and we were doing various things to add functions.

BG: And this was the non-commercial version of INGRES you were using?

MS: Right. But it was clear that this was not a viable long-term strategy. At some point you had to bite the bullet, throw everything away, and rewrite it. So in 1983 we decided to throw everything overboard and start writing a new database system called Postgres. The key goal of Postgres was extendibility, which is to make it work on GIS [Geographic Information Systems], make it work on document management, make it work on CAD. By then people had put in referential integrity and the notion of triggers, so there were these ad hoc ways of fixing up the database system. They put in a much more general rule system. And then the third thing, which I wish actually had gone somewhere, was you hear lots of people say if you look at double entry accounting they don’t ever overwrite anything, you put in corrections. Relational databases have this notion that you want to update something you overwrite the old value. So I said gee, it would be fun to build a system that didn't do that, that kept the old values so that you could then rewind time and do time travel. So Postgres did all of that. It was a very, very innovative system that we worked on throughout the mid 1980s at Berkeley.

BG: Why weren't you doing that as part of or in conjunction with Ingres?

MS: Well, I say they did eventually put in abstract data types.

BG: But that was later, I'm saying during that period of time, why did you do that as part of the university, as against doing it as part of the company?

MS: Well, because the company was focused on things that would generate revenue in the near term, and these were sort of science projects.

BG: Did you ever propose to the company that they underwrite these research projects, or to the VC’s, or to anyone?

MS: Well, we were well funded by the various agencies of the federal government. And I say one of the frustrations was my realization that they weren't. I guess the simple answer is that it would have been very hard for INGRES to morph into Postgres, you would've had to rewrite everything, and for them to do an incompatible transition from the old product to a new product is just really, really difficult.

BG: That's of course where you were discussing literally eating your own children.

MS: Yes they would've had to eat their own children, and it was clear that in the brutal competitive environment they were in, that they had no stomach for that. And also they didn't spend any money on research; they much preferred us to get funded. They viewed us as their research department that they didn't have to pay for.

BG: So you felt there was no conflict of interest in doing this?

MS: No.

BG: Who were you working with on Postgres?

MS: That was me and Larry Rowe.

BG: Gene Wong was not a part of that process?

MS: Gene wasn't, he wasn't involved.

BG: Were there any other people who were key to the work you were doing on Postgres or was it just the two of you?

MS: It was just me and Larry.

BG: What roles did you play in that?

MS: I think we were co-leading the project.

BG: Did you have programmers or graduate students doing the programming?

MS: For this whole time we had one fulltime programmer and three or four or five students.

BG: It's incredible what relatively small teams, skunk-work-like groups, can really accomplish isn't it?

MS: One thing that isn't very well known, well, maybe not as well known, is that to a first approximation Bill Joy wrote BSD, pretty much. I mean there were other people involved, but a huge piece of the system was him. You find a couple of super programmers and they can move mountains. We had our own machine at the time when this was not very usual, and we would give students a terminal to take home, so they could run night and day on their own machine. Eric Allman, who's now fairly well known as the inventor of Sendmail, was one of the Ingres programmers who wrote just huge amounts of stuff. We'd look for super-talented people and we were attractive because we had a playpen that was better than what was available. Throughout most of this time a small team could produce huge amounts of code.

BG: I think that's a very perceptive thing.

MS: The other thing that you do is that this is straight rapid prototyping, so there's no document. You don't write functional specs or any of the stuff that people do. You just go and do it, and see if it works, and if it doesn't work, well then you try something else. So, with a small very, very bright

team you can get a lot done.

BG: Yes, the argument is that the productivity ratio of the really top programmers versus the average programmer is ten to a hundred to one. It's an unbelievable ratio.

MS: That's right.

BG: I've had a few of those I've had the pleasure of working with in my life and it's really just sort of you light up when you find people like that.

MS: Postgres SQL, which is a popular open source system, is basically what we wrote with a small team in the late 1980s and early 1990s at Berkeley.

BG: Before I forget, going back, when did Ingres support SQL?

MS: Within twelve to eighteen months from the IBM announcement.

BG: But that never enabled Ingres to catch up on the gap that Oracle had opened up by that point in time.

MS: That's right.

BG: Okay. Now back to Postgres. You're doing the work on that in the mid 1980s, but Illustra is formed at some point in time. When does that get formed?

Illustra formed to sell Postgres

MS: Illustra I think was formed in 1992.

BG: That's quite a long time from the time you first did Postgres. You were doing that in the mid 198s, but you don't form the company until 1992. Can you tell us a little bit about what happened there and what the timing was?

MS: Until I left Ingres it would've been a conflict of interest, and I hadn't really written off Ingres until a year after the Ask acquisition, and so it took a while to realize that that was not going to be a commercialization vehicle for the Postgres ideas.

BG: You were distributing Postgres similar to the way you had distributed INGRES earlier?

MS: Yes.

BG: And you had a number of people who were using it, but was it all on an open source kind of thing at that point in time?

MS: It was the same model as INGRES.

BG: Same model exactly, okay. So now you leave Ingres after it's bought by Ask, some time later.

MS: It was more than a year later.

BG: And now you've got Postgres that you and Larry had developed; who else decides to form Illustra with you?

MS: The two of us.

BG: Just the two of you. Did you go and get VC money again?

MS: You're not really fundable unless you have a fulltime executive, so we needed the equivalent of Jon Nackerud to be viable. So the way Postgres was funded was that by that time Gary Morgenthaler had become a VC, and so I approached Gary. I said, "How do we get this off the ground?" and he said "Well, I'll run it and Morgenthaler Ventures will fund it." So that's the way Illustra got off the ground.

BG: The relationship with Gary while he had been running Ingres was a positive relationship?

MS: Yes. He's a very smart guy.

BG: Yes. There was nothing that had gone sour as far as that relationship was concerned. We've found in a lot of cases that founders and professional managers who come in don't match up well, and that the professional managers displace, or try to displace the founders. That's not uncommon. That didn't happen here though.

MS: Didn't happen. Gary has two features: one is he's very, very smart. I've met a lot of professional managers who are just idiots; that would create a really bad dynamic right away. Gary is smart. Then the second thing was his belief to keep us involved in the decision-making process, which I really appreciated.

BG: Very wise on his part I think.

MS: He's a very good guy, and you have interviewed him I take it?

BG: Oh yes, we have a couple of hours of his oral history. Okay, now you form Illustra. Again, the one day a week kind of thing effectively, and Gary brings in professional management people.

MS: Yes.

BG: Did you rebuild Postgres for Illustra, or were you able to use the open source as a base?

MS: My firm belief is that if you want to start a company and get something out quickly at low cost, what you do is take your open source project and say "Make this fast." Because the minute you say to rewrite everything, they'll spend two years sort of scratching their navels thinking about what to do. So you just don't give them that option, you say "Make this work." And so we handed them the open source Postgres code and said "Make this fast."

BG: What platform were you on with Postgres? Were you on VMS or a whole range of platforms?

MS: Well, Postgres ran on UNIX. I mean the university prototypes solely ran on UNIX, and so the initial version of Illustra was on UNIX.

BG: And by that point in time there was a sufficient UNIX market to make it worthwhile from a marketing standpoint?

MS: Correct.

BG: Did you use some of the same programmers you had used at the university or did you hire a new programming team?

MS: What happened in each case was that you got a couple of the university team to move over to the company, but it was mostly new people.

BG: Where did you locate that operation physically?

MS: Both of these companies were in Oakland, California.

BG: Not in a basement this time?

MS: No, no, we had enough money. This time we had more money and also Gary Morgenthaler didn't have an appropriate basement.

Problems with Illustra

BG: Let's continue with the story of Illustra. So now you've formed the company. You continue to play the CTO role. Gary Morgenthaler is the chief executive. As a business, what happens over the next few years?

MS: We're basically selling a next generation system, and if you're willing to use the extensions, they just result in blindingly better performance. But there are two problems. First of all, the extensions are not standard, and so that's an impediment. And secondly, in the case of, for instance, GIS, you need polygons, lines, line groups and the appropriate indexing optimizations, all of which Postgres had and Illustra had. Though we found that people wanted the ones from Arc Info or elsewhere, they wanted to get extensions from major application vendors. We spent our time selling to visionaries, and I'm trying to sell the application vendors on why they should morph their applications to take the pieces that should go inside their code and put them inside the database system. The app vendors would hound us and say, "How many customers do you have, and explain to me why it's a great distribution channel for me?" So we found that it was difficult to get partners because we couldn’t produce what the customers wanted. We were having a hard time convincing major vendors to accept converting their architecture to use Illustra: "We're doing just fine on the old system, explain to me why this is going to help sales."

BG: Did the customers need to have this product in addition?

MS: No.

BG: So this was stand alone.

MS: Yes. Things were tough. What happened was that the Internet took off, and the VP of Marketing at the time, a guy named Bruce Golden, handled this really skillfully. Illustra became able to do images, exactly what the Internet supported. So the Internet wave is the reason why we were a good database. It was graphics and text so you could correlate them with other stuff.

BG: Who are your buyers? Who was buying Illustra at that point?

MS: At the time Illustra decided to sell Postgres, which I think was 1996, we had revenue of about $5 million a year. You had all these big elephants saying, "Oh my god, we've got to somehow figure out what it means." Everything was just showering money on Internet projects to try and figure stuff out. So our customers were largely enterprises trying to figure out where you're at.

Sale of Illustra

BG: So who bought the Company?

MS: The big deal was, we were having a hard time getting going. If you don't generate enough revenue, then the app vendors say, "Why should I bother?" That was number one. The second problem, which was easier, was that Postgres was built to be this fancy extendable database system. It wasn't built to do bread and butter online transaction processing, so it was slow. Within any fancy application, there was a bread and butter business processing piece, and Illustra was no good at that piece. So those were the two fundamental issues that Illustra faced. Then what happened was a benchmark from Netscape, which at the time was a big Informix customer. They were thrilled with Illustra and said, "We love it." So Illustra almost took Netscape business away from Informix. That got Informix's attention and caused them to be willing to buy us.

BG: Was Roger Sippl still there or had he gone. Do you remember who was running Informix?

MS: Phil White. From Informix's point of view, it was a forward looking technology. From Illustra's point of view, it solved both of the problems that were bedeviling us. It gives us access to an engine, has good OLTP performance, and a distribution channel with some throw weight. So that was the reason for the merger with Informix.

BG: Was that a cash deal, or was that stock?

MS: That was a stock deal.

Working at Informix

BG: And you stayed with Informix for a period of time then.

MS: I inherited the job of CTO of Informix. I was a big company employee for a while.

BG: You stayed with Informix until 2000?

MS: Yes.

BG: How did that work?

MS: It didn't go very well. I think what happened in the acquisition was, of course, the executives largely get heaved overboard. The sales force at a startup is a bunch of very smart technical gunslingers, and they didn't blend with the button downs, sell to the enterprise sales force of Informix, so they largely didn't do very well. Informix had a completely different engineering culture than Illustra had. In their point of view, the way things would work was to integrate Illustra into Informix, and by and large that wasn't very good for the employees. I personally found myself relatively ineffective.

BG: But you stayed on for four years.

MS: It takes a while to realize that you're not effective.

BG: What was happening to Informix though? They had some problems during that period.

MS: They sure did. I think in 1997, Informix had a serious down quarter, followed by a re- statement of revenue, or followed by the board firing the president, followed by a re-statement of revenue, followed by a new president, followed by three years of turmoil and then I left.

BG: I'm just surprised you stayed involved that long. It must have been very unsatisfying.

MS: Well, the trouble was that the technical strategy of Informix at the time was taking Illustra functionality and to put it into their high performance OLTP engine, which would create the blend. First of all, that took about two years to get it done, which was a lot longer than I expected. And then it took yet longer to get that engine to where it was the OLTP engine of Informix, so I felt like I had to see through that process.

BG: Were they in fact able to integrate those two into a single engine?

MS: Yes. It was just massively expensive and took a long time.

BG: But at the end, was the performance satisfactory when it was finished?

MS: Yes. Informix Version 9 was a fabulous system technically.

BG: What was happening to your competitors at this point in time?

MS: What happened was, at the time of the acquisition, Informix immediately started a bunch of word wars with Oracle and they made a huge amount of noise. Let's see if I can say this in a way that

doesn’t get me sued. I think that what ended up happening was that Informix had been used to a 50 percent year over year growth rate, and largely they were pushing intra query parallelism because they had it, and Oracle didn't, and they were sort of fine in the early 1990s. What happened was, Oracle got intra query parallelism, and then number two, it seemed like about 1994, both the rate of growth of the software companies, of the whole industry, just slowed down, everybody slowed down, the growth rate at Informix slowed down. But Phil White continued to push the company to grow at 50 percent, and eventually your string runs out and that is what happened when he had a down quarter, I think in 1997. So the minute you had a down quarter, then the financial viability gets called into question. You start having grumblings among the executives, just as the Informix product became difficult to sell. And as I say, this was a real education for me. I had no idea how hard these events are to recover from. So anyway, I stayed with Informix through Phil White's tenure as well as that of his replacement. When the board hired the next CEO I said, "I've had enough, I'm not up to trying educate another CEO." I said, "I'm out of here."

Competitive Climate

BG: Two quick questions. During this period of time Sybase was now in existence. Were they a significant competitor to you? What were they doing to compete with the extended functions that you introduced through Illustra and Postgres?

MS: Sybase is peddling itself as an OLTP engine.

BG: So they're going for performance application?

MS: But they're not effective. We were competing against Oracle really.

BG: The other companies continue to be competing heavily in the mainframe spaces that you're not working in. They're really not a major factor. Is that a correct statement?

MS: Yes.

BG: Was the IBM AS/400 a factor here?

MS: No. I think that's really interesting, because IBM was a real lightweight in the UNIX market. That didn't have to be the case.

BG: Was it the RS/6000 or whatever their machine was that they put in as a RISC machine a factor? That was their only real significant entry in the UNIX market, wasn't it?

MS: Yes, but I think they were never serious about competing for UNIX. You'd have to ask somebody why that was the case.

BG: In contrast, the AS/ 400 was an incredibly successful machine that IBM put out in the business markets.

MS: And it had good business functionality, and terminals, their block mode and all that stuff.

BG: IBM knew how to do that, and they finally pushed it magnificently. In contrast, in the RISC market, they were just one of the four or five major players, and they were late there. They had the technology first, and yet HP and the others all beat them to market.

MS: I think this is a case where, in the business market, they really understood the market, because the AS/ 400 doesn't look anything like UNIX. UNIX was sold to the scientific and research community and then morphed into everything else, and they certainly had no technological advantage against anybody else.

Technology Transfer

BG: I think you raise a question I'd like to explore a little bit here. The technology transfer issue and the significance of technology is different in the business area in general. There are different kinds of applications and they seem to be more competitive than the technical markets. You were selling to different kinds of people than the typical financial clients and corporate vice presidents. Is that an accurate statement?

MS: I actually disagree. I think you've almost certainly read a book called "The World is Flat" by Thomas Friedman. It seems to me, the US is in a global war with everybody else, but primarily China. It seems to me we have two big advantages. Number one, we have the best research talent in the world by far; and number two, we have the best VC system in the world by far. So I think the reason the US stays competitive in IT is that researchers come up with neat stuff. The VCs fund it and create startups. Then either the startups get bought by the elephants, and thereby you do technology transfer, or people like us are successful as independent companies and generate whole new markets. I think there's no question that that happens in essentially all markets. I think, for example, in the case of tick stores—let’s go straight to Wall Street. Every company wants to store a multi-year history of all ticks, and that's a huge amount of data that they want blindingly fast, query response time to. Relational systems are essentially not used on Wall Street. In fact, they're using very special-purpose stuff. One of the companies I'm now involved in has a warehouse-oriented stock-tick store. The ticker guys love because it does fast queries on their sort of stuff.

BG: Is this Vertica Systems?

MS: Right. So this new idea is an example of how technology transfer happens just pretty much universally.

BG: I was saying something different. It appears that the big companies have a great deal of difficulty, even if they have big research organizations like IBM, in getting that research out of there and into their product divisions. That was the point I was trying to get to.

MS: That's absolutely true. Why they can't do a better job, I think is this. I've interacted with the CIO at one of the large investment banks in Boston, and he runs a large IT shop, which is very, very

cautious: belts, suspenders, ropes. They are really adverse to new technology, because the whole culture is run by a bunch of bankers. He's trying desperately to figure out a way to infuse new stuff into his organization, just because the current stuff is so old. Over time, the gunslingers leave, and you're left with people who are plodders, and then the downside of screwing up is so high, and the upside of succeeding is so small, that you get to where there's very little innovation.

BG: Yes. I think this is the point that you've made in some of your interviews; start ups seem to be the American way of getting the new technologies developed, and then they are bought up by the elephants, and destroyed in many cases.

MS: You get the technology transfer to happen. I'm just saying, instead of it not happening at all.

BG: It does happen.

MS: It does happen. The Internet is an American phenomenon, and Google and Yahoo and Amazon are American companies.

BG: This story has been true in the IT field since the 1980's and before.

MS: I'm just saying, suppose some major innovation, web 3.0, whatever it happens to be, comes from China. Then it's entirely possible that the major companies that spin out of that new transition could be Chinese. The only reason that American IT remains strong, it is the dominant force in the world, is that this technology transfer is happening here and not somewhere else.

Mariposa and Cohera

BG: The Computer History Museum has been considering a major exhibit. Among the themes that are being considered are just these points. It's the technology advances and the money, the entrepreneurship, that's unique here in the United States, has really forced growth in the IT industry.

Let me go back. I'd like to continue along our chronological trail, and then I have some other things I'd like to cover. So Informix: you're leaving there. What's happening at the university? You have some new things you've been working at during the 1990s at the university with government money and other sources of funding?

MS: We built an innovative federated database system called Mariposa.

BG: What is a federated database system?

MS: It takes databases written by multiple entities and gives you a logical view that they're one.

BG: And that was called Mariposa. And you were involved in a similar kind of mode that you had used with INGRES and Postgres?

MS: Yes.

BG: Did you have a partner or someone that you were working with particularly in that case, or was that pretty much your stuff?

MS: That was just me.

BG: And that was in the 1990s you were doing this?

MS: So that was in the 1990s.

BG: Any other major things that you were working on during that period.

MS: That was my major focus, and that turned into a company called Cohera.

BG: And the timing on that is about 1997?

MS: Sounds about right.

BG: You showed that as being founded then and you continued as CTO until 2001. You obtained VC money again; did Gary [Morgenthaler] help out with this one again, or someone else?

MS: That turned out to be funded by Sequoia Capital. Morgenthaler was not involved.

BG: You've gone to a series of different VCs. I guess by this point you have a hell of a lot of credibility, is what it sounds like to me.

MS: I know them all.

BG: And they know you've succeeded before.

MS: That's right.

BG: I think that makes a great deal of difference.

MS: Absolutely, so I'm very fundable to do most anything.

BG: And they recognize that you're willing to have a CEO run the company while you have the CTO role. What happens with Cohera?

MS: Cohera took Mariposa and built a beautiful distributed database system. To a first approximation, nobody wanted it. It was a big lesson in data integration. The big problem is that if you're Barnes and Noble and I'm Amazon, and you want to integrate our two schemas, these would have been developed independently. And the chance that they mean the same thing is zero. So for instance, in a more business data processing vein, if you're the human resources guy in Paris, and I'm the human resources guy in New York, well we both have employees and employees all have salaries. So the US salaries are gross in dollars. The salaries in Paris are net. They include a lunch allowance and they're in Euros. So converting between them is really hard. So distributed

databases run up against semantic heterogeneity, and they don't have a good solution to that. Semantic heterogeneity is just a very, very hard problem, one I'm continuing to work on.

BG: And we're not just talking language difference. We're talking definitional.

MS: The language issues are the easiest ones to deal with. It's the semantics of what the bits actually mean. They're often very difficult to convert.

BG: It's one of the most difficult things in a requirements study: what exactly does this person mean? Within a company, that tends to be relatively common. But between two different companies, or two different parts of the same company, you get very different definitions, don't you.

MS: Look at the data warehouse market. The idea is that you take operational systems, you construct a common warehouse schema, and then ETL does this mapping. And ETL in the real world has proved to be infinitely difficult for exactly these reasons. You're just trying to transfer data in bulk, and you get stuck on exactly this semantics problem. Federated database systems are trying to do it in real time, which just makes it that much harder. It's just a big issue. The thing I find fascinating is that Microsoft is out touting service oriented architectures and web services. Imagine you're the HR guy. You put up your stuff as a web service. I'm the HR guy in New York. I put up my stuff as a web service. If people query both of our databases using some fancy web service interface, they're not going to be any better off, because your stuff doesn't mean the same as my stuff does. So the elephants are touting web services as a panacea to integration, but that isn't true. So I think it remains a thorny, difficult problem.

BG: What happens with Cohera at the end? Do you close down, do you sell, what do you do?

MS: It was actually sold to PeopleSoft at very low valuation.

BG: That did not make you wealthy.

MS: No. I didn't afford this house based on Cohera! That was not one of my more successful ventures.

Change at Berkeley

BG: So a big change in your life takes place prior to that period of time. In 1994 you become a professor at the graduate school in 1994. What's going on?

MS: That's a euphemism for what happened. The University of California is publicly funded by the state. So the state has a financial crisis every so often, and if the state has a financial crisis, so does the university. So the university had a financial crisis in 1993 or 1994, and 85 percent of the costs are salary. So the only way to deal with such a crisis is to get rid of people. So Berkeley, or the University of California, also has the interesting characteristic; if you started in the 1970s, you could opt out of social security, and instead put the same amount of money into a private version of it run by the university. So the university had its own version of social security. So the retirement fund is way overfunded, because it's a pay as you go kind of system. It has good money managers, they've done well, and university faculty member don't ever retire. They die at their desk. So you have one pot of money that's ridiculously overfunded, and another pot that's

empty. So of course, standard bureaucratic wisdom is, you just bribe people to retire. So basically, in 1994, that was the first year I was 50 and qualified, so I said, "Fine, I'll take the package." So basically I retired, and professor of the graduate school is a euphemism for that.

BG: You were a very young emeritus though, weren't you?

MS: I think what ended up happening, at the 1994 crisis, there were 11 computer science faculty at Berkeley that were eligible, and seven took the package, because we were all employable. And the package was basically, if you're going to retire any time in the next 15 years, they made it so you might as well retire now. So I say, "Fine, I'll go and do something else."

BG: But you maintained an office there and so forth?

MS: Yes. I maintained an office, but I was drawing a pension from the retirement system instead of drawing a salary. And I only taught if I wanted to. You could then do anything you wanted to. Starting in 1994, I was a pensioner, and doing other things, and continuing to work one day a week for Informix, and one day a week for Cohera.

BG: And they were paying you salaries, I assume, for working for those companies.

MS: I also had consulting income. So I was well paid.

BG: Did you do any other things in those last few years at UC Berkeley, other than Cohera?

MS: The other thing that sort of bothered me is that until very recently, user interfaces to relational databases have been very primitive. I was trying to build a fancy viz system, that would be database driven. I guess the closest thing today would be something like Spotfire. We had a very innovative viz system.

BG: You were working on that while you were still in California?

MS: Yes. That was what I was working on while Cohera was doing its thing, and while I was CTO of Informix.

Moving to the east coast and MIT

BG: What made you decide to move to the east coast?

MS: I'll give you a couple of different answers. One answer is that my wife is from the east and has a huge extended family in the Boston area, and had been lobbying, increasingly aggressively in recent years, to change coasts. And so at some point, in the interests of staying married, it was time to live where she wanted to live for a while. So that's one answer. Another answer is, there's never a good time to move, and at some point, why not try something else? Another answer is that my wife wanted a bigger house, and looking at trading a million dollar house for a million and a half dollar house in the Bay area is not my favorite thing in the world to do. So if we moved, we could buy a bigger house for next to nothing here, or what we think of as next to

nothing. So those are all various pieces of the answer.

BG: Fill in just a little bit though. You got married when?

MS: To my second wife in 1984.

BG: And you have two children?

MS: We have two children.

BG: Did you approach MIT or did they approach you? What happened?

MS: We decided to move here and we did move here. She has a large extended family, as I said, south of Boston. So to a first approximation, I put a pin in the map, drew a circle around the pin and said, "Outside this circle." New Hampshire is especially advantageous tax wise, and since I grew up here, it's nice to look at birch trees. So we decided to move to New Hampshire and looked around for a community with good schools. So that was pretty much the decision making. We chose initially to live in Bedford, which is outside of Manchester, and then moved into Manchester a couple of years ago.

BG: And you have a house on Lake Winnipesaukee?

MS: Well actually, when we moved here, I lobbied hard to live here full time and my wife would have none of it, because it's too far from her relatives, and in the winter, it's pretty desolate here. So the basic agreement is, as you well know, if you move from California to here, you basically trade two for one. So we could buy two houses here for one house there. So we bought both and we've had a vacation house since we moved here.

BG: Let's move ahead, back to the chronology. We're going to go much more briefly because history has to do with things at least ten years old, and we're now getting within that ten year range. You move here, you don’t have another company at that point, you start something called Required Technology, what is that to do with, or what does the name mean?

MS: We moved here, and I naively figured I'll just be here, and I'll be able to network and sort of build up the contacts I had on the east coast quickly, and of course that didn't happen. And so I went around to all the local universities, all of whom would be happy to have a relationship with me.

BG: Of course.

MS: And so I decided to join MIT. In the long run it's the right thing to do, but in the short run they had no database expertise whatsoever. So at MIT you had zero faculty in databases, zero courses, zero students. At the beginning, you have a stationary flywheel that you have to start to get moving, so that's taking a while. In the meantime it was clear the only way to get anything done was to form an amalgam of all the other universities in the area, all of whom had one or two database faculty. So, the whole Boston area acts almost as a single research group. Among all the universities there's a huge talent pool and so I pretty much have organized that, and that's been

very successful.

BG: Tell me about it.

MS: Brandeis, Brown, Northeastern, Worcester Polytech, MIT, Harvard and the University of Connecticut in Storrs are all within an hour of each other (except for UConn), and they all have database faculty. And so the eight or nine of us at those universities pretty much act like a single research group. We have a single colloquium series; we have joint projects among all the universities, and it's a way to get to critical mass quickly. And over that period of time MIT has hired a database guy, we're starting to get database students, we're starting to get MIT-only projects, but it's just taken a while. There's a lot of collaboration among the Boston area universities. MIT is a weird duck in that if you want to join MIT, space is controlled by the research labs and not by individual departments, so you join both a department and a research lab, and you've an academic appointment from the department but you get an office from the research lab. CSAIL is the main computer science research lab at MIT, and is the joining of the laboratory for computer science and the AI [artificial intelligence] lab, which we merged a couple of years ago into the Computer Science and Artificial Intelligence Lab. So that's what CSAIL is, and that's the research lab I'm officially associated with at MIT.

BG: The other thing you mentioned, the database collaboration, is an ad hoc structure; it's not a formal organization?

New Ventures

MS: It's an ad hoc group. And that's one of the big advantages of being in the Boston area. There are a gazillion universities within close proximity and the union of them forms a powerful, powerful entity. So, now I can talk about Required Technology. Out of the blue I got a call from this guy saying "Do you want to consult for a startup in New York City?" and so it was a startup that I was just helping out.

I'm wandering around among universities and Stan Zdonik, who's at Brown, is somebody I talk to a lot, and we decided that there was going to be this fire hose of streaming data, and I said, "Well, there's going to be a revolution in sensor networks," and so the plummeting cost of sensor technology says that everything on the planet of significance is going to get sensor tagged and report its location or state in real time, and that's going to create an avalanche of down-stream streaming data that current system software is very ill equipped to deal with. So, we started this project called Aurora, which was sort of between Brown and Brandeis and MIT, to build an engine that would process streaming data blindingly fast. And since we are database guys you want to invent a version of SQL that will work on real time data as it goes by on the wire. And again we built a prototype, it was government funded, and it was public domain, and we spun it off into a company called StreamBase, that's here in Lexington, and they're doing just fine, so I work for them.

BG: Any VC money?

MS: We used east coast VC's, so we used Bessemer Ventures and Highland Capital Partners.

BG: And so that's currently an ongoing thing with you. Again, you're playing the CTO role?

MS: Yes.

BG: Okay, and CSTORE, you've mentioned something about that before.

MS: One of the things I often wonder about is how I come up with creative ideas, because I certainly don't get them sitting up here and looking at the lake. I get it by interacting with smart people. If you look at streaming data the basic message is that an SQL-oriented system is a good idea, but the traditional ones don't work, and you're just smoking something if you say put a hundred thousand messages a second that you want to process in real time into a database;, I mean, that just is not going to work. So, you say, "Well, I wonder where else there's an opportunity that says one size doesn't fit all," which translates into: "is there another case where SQL is a good idea but that the engines that currently exist from the elephants are not a good idea." I'd had some exposure to the data warehouse market both at Informix and at Required, and so over a couple of years there unfolded a collection of ideas on how to make warehouse data go fast. And collectively you can think of this as column stores as opposed to row stores. System R and INGRES and all the other systems have been row stores, which means the next thing in storage is the next attribute over in the same row. And that works well on OLTP [on line transaction processing], but it turns out for a variety of technical reasons that it's a lousy idea on data warehouses, and you're much better off having the next thing in storage be in the same column in the next row, which is just turning everything ninety degrees. So CSTORE was a research prototype at MIT that's a column store as opposed to a row store. We built enough of it that we could get processing speeds, and it's dramatically faster than a row store, like a factor of fifty. And so based on that, since I'm now getting older, I said, "Well, instead of making this really work on our own, let's just get VC funding right away."

BG: You're changing your model?

MS: To build it at MIT would require big sums of money, and it's easier to raise money from Sand Hill Road [Menlo Park, California], or from the east coast VC's, than it is to raise it from the government. So you might as well let the VC's use their nickel as opposed to the harder route of getting money from the feds.

BG: I think we come back to the fact that it's your credibility that makes that feasible.

MS: That's true, absolutely true. So anyway, Vertica is a column store. In the same way that StreamBase is a factor of fifty faster than a relational database system on streaming data, Vertica is a factor of fifty times faster than a relational database on data warehouses. But the world doesn't stop there, so more recently I've been looking at OLTP, and we have an upcoming paper in VLDB in Vienna next month that goes through a prototype called HSTORE, which is an OLTP engine, and it's a factor of eighty faster than the elephants on the TPP - C which is kind of standard OLTP benchmarking.

BG: What does the H stand for in this case?

MS: Horizontal. It actually is a row store. As another example, in the scientific market most scientists have arrays, and those are devilish to simulate on relational databases, and if you actually have an array store you can beat the elephants by a factor of fifty or a hundred.

BG: And you have n dimensionality in the arrays, or just three?

MS: n dimensional. So in every market I look at that has any significant size, if you specialize the architecture you can win by a factor of fifty or a hundred.

BG: And you're talking about software architecture?

MS: Software, software only.

BG: Wow!

MS: So, I think where I see the relational database guys now is they're selling thirty year old architectures built in an era where business data processing was the only thing you were trying to do, and they're now trying to extend that architecture to do everything, and in the process they turn out to be good at nothing. My current complete focus is to build specialized engines that are vertical market specific, that just run circles around the elephants.

BG: You've led us very nicely into the whole series of subject areas that you've mentioned in some of your interviews on these specialized areas. I'd like to talk to you about some of those if I could.

MS: Sure.

Other Specialty Models

BG: You talked about Text Searches. Does this approach have meaning for Text Searches as well?

MS: Well, it turns out if you look at Google or Yahoo, none of those guys use relational databases because they're too slow, so it's an area where there's yet another specialized engine that has a hundred percent of the market.

BG: Let me back off just a little bit. We have mechanisms and tools for dealing with Tabular processing. You have languages to query those things when we think of them in that form. Is there a similar set of things in the array world?

MS: Well, I'll give a couple of different answers. One answer is in the case of streaming data you simply want to do Stream SQL, which is SQL with some stuff added that makes it work on streams. So I'm not opposed to SQL as long as there's some other engine underneath it, and as long as you put in the stuff that you need to make it work in whatever your vertical market is. So that's one answer: preserve SQL as an intergalactic interface, with various extensions for these various special markets and very different implementation engines underneath. The second answer is the text guys have no use for SQL, and their Google-style interfaces are what everybody uses. That is the intergalactic standard. So you can say personally that SQL buys you

nothing in that market. In the scientific market in fact the science guys like the Mat Lab user interface, which has nothing to do with SQL. I think in those markets where there is some other user interface which is dominant already, you should just use that and have your engine directly process that stuff, without SQL as an intermediary. I think it varies from market to market.

BG: You talk about latency and the problem of latency and how that slows everything down, the multiple steps of going into, and the timing issues, all these things you say are really critical in some of these applications, like the real time applications.

MS: In streaming markets, latency is everything.

BG: But you also mentioned that in the financial marketplace, if I'm trying to see what's happening because I'm an arbitrager or something like that, I need a different model.

MS: If you look at electronic trading, which is absolutely taking over, a feed comes out of the wall, and you want it. The electronic trading engine computes some sort of secret sauce which no one will say what it is, but it's basically real time business analytics; for instance, compute the momentum of IBM on a tick by tick basis, compare it with the momentum of Oracle, if the spread is big enough one way or the other, short one and long the other. So it's that sort of stuff. And the tolerable latency between when a tick comes out of the wall and when the trade actually happens is critical.

BG: The trade is actually recorded?

MS: Yes. When your window for arbitraging interesting events is right now, it's maybe fifty milliseconds going down to one millisecond, and so there's this latency war going on. All the people who do this seriously are getting direct market feeds because the consolidators like Reuters and Bloomberg introduce too much latency. Speed is everything, and if you want to put something in a relational database it takes you a second, and that's forever on Wall Street.

BG: So in many senses you can't afford to put things into a database.

MS: That's right, you cannot. You've got to do on-the-wire processing.

BG: You've got to deal with it as it is happening.

MS: That's right, and that's one of the big differences: the whole architecture of relational databases is so inappropriate for real time stream processing. The interface is fine, the implementation stinks. In the case of text, the interface doesn't work either, and the implementation doesn't work. So both characteristics are true. Now put more generally, one of the things we haven't discussed at all, which is very significant, is that in the mid 1970s there was a huge argument in the relational database research community that said should we invent a sub-language, like SQL or QUEL, or should we just extend the programming language like PL/1 with database stuff. There were proposals from both camps. Chris Date wrote an extension to PL/1, and a guy named Joachim Schmidt did Pascal R which was some relational extensions to Pascal. The sub-language camp won, but it's created things like ODBC and JDBC, which are devilishly difficult to use. If you want to run a single query from JDBC or ODBC there's a bunch of code; there's one SQL

statement and then all this junk to get stuff in and out and set up cursors and interface issues to the programming language. I think the right thing to do is to do language extensions that are language specific, because that gives the programmer a much easier task. Ruby On Rails is just one example of a recent system that does this. It has database constructs that are embedded in the language, so it just makes it much, much easier to program.

BG: Let's stay with that for a moment. You speak to the need for better programming languages, but one of the comments that I've heard over the years is that we can see the improvement in price performance in semi-conductors, we can see it in database storage devices, but we've never seen anything comparable to that in the programming world.

MS: Yes. In thirty years there's maybe been a factor of two.

BG: I mean looking at programming from 1954 on, and they tell me that there's not been a dramatic change in the design capabilities or in the programming capabilities. It's better, but it's not a hell of a lot better.

MS: That's right.

BG: Are there things that you can see in the way of development tools -- ways of changing how programs are designed and developed that would make significant ten-to-one changes there, instead of the two to three-to-one which is about all we've achieved?

MS: One simple answer is that in the business data processing world it's conceivable that relational databases are the right answer, although I want to come back to that in a second. In most of the other markets that we've talked about, relational databases are not particularly the right answer at all, and you'd be better off with a different data model. So I think expunging from our thinking one size fits all would allow various data models to flourish that would be easier to program against, because rather than having to convert your object structures from whatever you think of them, to tables, which is challenging…

BG: Or to arrays for that matter.

MS: Yes. Directly executing the user's model would be a good idea. In business data processing, there was a huge debate in the 1970s of relational versus CODASYL versus hierarchical data structures. But there are a couple of things that have happened in the last thirty years in business data processing that are really significant. One is that the end-user has moved from a specialist keyboarder to some schlep on the web, and the second thing is that every OLTP shop I talk to wants seven times twenty-four times three sixty-five, never go down, Tandem-style non-stop. Have enough redundancy so that there's only one state of the database system and that's up. Those two things alter the way you think about transactions a lot, and part of this is in my paper in VLDB. That causes you to revisit whether relations are in fact the right answer, and my guess is probably not. My prediction is that the database market is in fact going to morph into some number of markets, three to six, each of which will have a different engine and perhaps a different data model, and certainly a different user interface.

BG: You made a comment earlier on that the Codd model rationalized databases for you. Is there some mathematical work or anything going on now that will rationalize this concept or structure that you're speaking to?

MS: I think just a simple example was Peter Chen in the 1970s worked on what's called the ER Model, and in fact the ER Model is wildly popular in database design tools, basically omnipresent. And in data warehouses it turns out everybody uses what are called snowflake schemas, or star schemas. Those are really better modeled by an ER Model than by tables. So we'd probably be better off moving to Peter Chen's ER Model for data warehouses.

BG: Does that provide a mathematical foundation? That was the point you made earlier.

MS: It's just CODASYL. There was nothing that says you can't do a clean network model, it's just CODASYL didn't do it.

BG: Well, one could argue that COBOL is not a very clean language in that regard as well.

MS: I know. You look at COBOL and you ask why would anybody want to use it.

BG: Harry Markowitz, who helped form CACI in the 1960s. I think he was working on that approach at IBM Research in the 1970s; he called it Entity-Attribute-Set, with some kind of Event that triggered the changes. That was different from Peter Chen's work though, I believe at that time.

MS: I'm not familiar with Markowitz's work so I can't really comment.

BG: The idea of having some overall structure that is multidimensional seems to be a different way of looking at it. But we, as people - as programmers - have trouble seeing things in multidimensional space.

MS: But the scientists all think that way.

BG: Exactly, but business people don't.

MS: Yes. I'm just saying I think one size doesn't fit all, and back when there was only a business data processing market you could argue that one size fits all, but the argument against that today is overwhelming.

BG: Let me carry it one more step. Do you see the possibility of creating programming development tools that would enable you to create these various specialized application solutions?

MS: Yes. I'm not really a programming language guy. All I do is sit here and whine at programming languages, and I wish I knew more about them, because I think that's a very interesting thing to work on.

BG: You made a point that these are going to have to be individualized solutions, but with hopefully some generalized tools that one can use. That implies some kind of a, and maybe the word "language" is the wrong word, but some mechanism to enable you to build those relatively

efficiently. And I was wondering whether you've done any work on that or have been working with any people who do that?

MS: No.

BG: Okay. Another general thing that you talked about, you defined the difference you felt between technological disruption versus evolutionary change. I think evolution may be my word, not yours. There have been some of those; you viewed RDBMS as being a disruptive change.

MS: Absolutely.

BG: Do you view what's happening now as being disruptive as well, the kind of things you're talking about?

MS: Absolutely, because I think the simple answer is that the relational elephants are selling one size fits all, and if you have five different engines, then engineering has a challenge trying to keep them in sync when there are common features. The sales guys who are typically smart enough only to take the customer to lunch have no idea what to sell; the marketing guys have a challenge positioning products. The whole way that the enterprise software is structured would have to change dramatically. There's technological disruption but I think there's company organization disruption, and so the elephants will have a very hard time coping with this sort of change.

BG: So you again see new startup companies as being the vehicle that will help to find solutions to these kinds of things?

MS: If you look at a company like EMC, buying companies as fast as they can, before their core business craters, because why will people spend forty thousand dollars on a terabyte of disk when you can buy it for a thousand dollars. I think the elephants may well buy companies and let their old business crater and basically morph into a new kind of structure. That could happen.

BG: But in the case that you've been involved with, every time the company's been acquired by someone else, they haven't taken advantage of what they acquired.

MS: I know. That's right. I find that fascinating, and I think the business guys all report that the majority of acquisitions are failures, and it seems to me that people continue to do it, to get technology transfer, but it works very poorly.

BG: Now you were making the point that this is the way to get the new technologies created, but the mechanism of transferring that into production seems to be working fairly sloppily even here in the United States.

MS: Yes, I think that's right.

BG: And how do you correct that?

MS: The executives in the Fortune 1000 have to get a lot more sophisticated technologically, because

if you look at people who are customers of IT, they're very slow moving and ponderous. IT is their competitive weapon but they're not very good at taking advantage of it. And you look at IT companies and they're moving glacially slowly at this point, and this has to get solved or we're going to get run over by the Chinese.

BG: The IT executives don't tend to become the heads of companies, and the financials become dominant. I'm afraid that the people like yourself who bring so much technology to bear, how do you get them to appreciate and to deal with the values of that technology?

MS: The other answer is to look at somebody like Google, which has a very different technology development model, is staying independent, and is being very successful. If you can, the answer is to stay independent, and only get acquired if there's no other choice.

BG: But most people want to cash in. That seems to be a major goal for many people, to cash in at some point in time, didn't you feel that way?

MS: No, never.

BG: You were never an instigator of being acquired?

MS: No, no, I think in the case of Ingres, they were losing in the marketplace and you might as well try something. Now in the case of Illustra, to solve the two problems that we talked about earlier would've taken years and millions of dollars that they didn't really have. So, yes, I viewed them as not having any choice. But I would never advocate getting acquired.

BG: So, you were not a pusher to make that happen in those cases, because cashing in for many people is very significant, or it seems to be.

MS: Yes. The thing you've got to realize is that for startups the major proponent of cashing in is the VC's, and acquisitions are almost always bad for the employees and very good for the VC's. But I think at least in my situation I have enough throw weight to stop an acquisition.

BG: You've been in an extremely strong technical position.

MS: Yes.

BG: And that's an unusual case again.

MS: Exactly, exactly.

BG: I just want to pursue a couple of more small areas, and then I think we'll draw to a close. You speak of the term "embedded databases" in one of your interviews, what did you mean by that?

Data Flow Models

MS: If we go back to streaming applications, the Wall Street case, the data comes out of the wire typically directly from the exchange, and the steps you have to do is you've got to get it into your

system, you tend to have to clean it and discard outliers; you then have to compute some business analytics to find events of interest, and then you have to get instructions to a trading engine, and execute a trade all within ten milliseconds. That's pretty hard, but now let me make it a little bit harder; you're looking for events of interest, and an event is of interest if the real time analytics look a certain way. But then in addition, if I've seen this event five times already today, then chances are it's not of interest, so I need some stats. And trading engines want more and more stats, but they still have ten milliseconds to execute. So, you cannot cross a machine boundary in ten milliseconds, you cannot even cross a process boundary to a different process on the same machine in ten milliseconds. So embedded means everything has to run within a single operating system process, because it's too much latency to go any place else. And so StreamBase has an embedded database system that runs in the same process, no process switch to get to it, no ODBC. Getting from an application to the database system in the elephants is a very heavy operation, and it's very heavy because you don't trust the application and you're not willing to run it in the same address space, so it's across the wire. You have to authenticate whoever's running the application, etcetera, etcetera, etcetera. And you just need very lightweight stuff if you want to run fast.

BG: I'm hearing the term "data flow," that you're talking about, is this a substitute for database in your mind, in many of these applications?

MS: This is a data flow. It is a data flow world in streaming data.

BG: And how about some of these others, are they also data flow worlds in real time applications as against database worlds?

MS: Yes, for all real time applications that care about latency. There's been a lot of press about RFID, and Wal-Mart is pushing RFID on the pallet level. So, if you're moving pallets in a warehouse, you have the forklift guy, and he's moving a pallet every couple of minutes. You don't care where the pallet is within latency of a minute or two. The number of pallets you're moving a minute is maybe five, so you're doing five messages a minute and you don't care within a minute where the thing is. In this case, run any technology you want, it doesn't much matter, anybody will be able to solve your problem. If you look at the options market and you're trying to do electronic trading on the options market that's a hundred thousand messages a second, and now you really care, and data flow is the only game in town, you cannot store the data, it's just not possible.

BG: So you can't even have time to create the database in that sense.

MS: No. The paradigm of store the data and then look at it won't work. You've got to process the bits as they go by on the wire.

BG: Science and intelligence databases and those kinds of applications, is that the same kind of thing, or is that a different problem there?

MS: That's really fascinating, because as near as I can tell, and again the spooks will not tell you exactly what they're doing, but there are a couple of different problems that they appear to be

working on. One is that they would like to put a video cam on every light pole in Baghdad, and watching real time stuff going by, so they want to do streaming analysis of vehicles as they go by. And then they want to have a database of everybody they've seen for the last n years: they want an enormous store of everybody they've seen. You want to discover the bad guys. And discovering the bad guys is partly figuring out that there's an interesting real time thing going on, and then correlating it with a massive historical database, all of that very fast. And so the problems those guys have are really technically challenging and they're a mix of data flow and massive data warehouses. And the other thing is that the spooks also appear to be watching some significant set of all the Internet traffic and all the phone traffic in the world, and they want to see who is communicating with whom, and basically build Facebook style networks of who is talking to whom. And then if they decide somebody's a bad guy they want to be able, in a hundred microseconds, to figure out everybody that's within two levels in this connection network. So again, massive amounts of data, very real time. And so the spooks have fascinating problems.

BG: The last thing and then we'll close, you also mentioned about the object oriented database and its failure of handling CAD, and I was wondering if you happened to think of a very short comment about that if you could?

MS: Sure, I think the trouble with CAD, the OODB [object oriented database] guys, their penultimate example market was CAD, and the trouble was having talked to AutoCAD and Mentor Graphics. People who are in the CAD business didn't want a new database, and so the customer wasn't interested. And so there's a fundamental failure that you had a new mousetrap and the guys you thought you were building it for didn't want it.

BG: They weren't going to catch a mouse that way.

MS: And the reasons were fairly straightforward. The problem was that if you're in the CAD business you have a representation on disk, you swizzle the data into main memory, you now have a massive amount of tools that massage it in main memory and when you get done you put it back. The only piece of the problem that the OODB guys solved was the swizzling back and forth, and that wasn't enough of the CAD guy's problem to be interesting. So then they just didn't solve a big enough piece of the problem to get successful.

BG: We're going to draw this to a close. Thank you very much Michael.