An Interview with

DONALD D. CHAMBERLIN

OH 329

Conducted by Philip L. Frana

On

3 October 2001

San Jose, California

Charles Babbage Institute
Center for the History of Information Processing
University of Minnesota, Minneapolis

Donald D. Chamberlin

3 October 2001

Abstract

Don Chamberlin is a research staff member at IBM Almaden Research Center in San Jose, California. In this oral history Chamberlin recounts his early life, his education at Harvey Mudd College and Stanford University, and his work on relational database technology. Chamberlin was a member of the System R research team and, with Ray Boyce, developed the SQL database language. Chamberlin also briefly discusses his more recent research on XML query languages.

This is an oral history with Don Chamberlin conducted by Philip L. Frana on October 3rd 2001, in San Jose, California, for the Charles Babbage Institute Center for the History of Information Processing. This interview is conducted under auspices of the National Science Foundation Software History Project.

Frana: Don, could I just have you introduce yourself and your current title at IBM?

Chamberlin: Sure. I am Don Chamberlin. I am a research staff member. Just about everybody in the Research Division of IBM is a research staff member. We have kind of a one-class society here, so I've been a research staff member since 1971, which means I've gone about a little over thirty years without a promotion. And that is pretty much the way I like it. It is a good job.

Frana: I noticed that this facility is high up on a hill, kind of isolated and by itself. Is that by design?

Chamberlin: Well, IBM has several research labs and they all tend to be in suburban locations. They are all very nice buildings. I joined IBM at the Research Lab in Yorktown Heights, New York, which is a very beautiful building. It was designed by Eero Saarinen and it's up on top of a hill with forest as far as you can see. IBM takes good care of its research people. It gives them nice places to work.

Frana: Yes, and, I suppose, thought provoking locations.

Chamberlin: Yes, I guess so.

Frana: Could you tell me a little bit about your early life and how you became an IBMer, particularly your education?

Chamberlin: Sure. I was born here in San Jose, California. I went to high school in Campbell, which is a little town that's now mostly surrounded by San Jose, but in those days it had a little character of its own. I went to Campbell High School. In 1957, when I was in the eighth grade, the Russian satellite Sputnik was launched. It became clear to my generation of people, growing up, that technology was an important thing and a national priority and there was a shortage of technical talent and this would be a good field to go into. That really influenced my early thinking about what I wanted to do. I always liked to tinker with things, and when I graduated from high school I went to a college that was almost brand new—Harvey Mudd College in Claremont, California. Harvey Mudd is a very small college that specializes in science and engineering. It was founded in 1955 and so I was in one of the early graduating classes from Harvey Mudd—probably class number six or thereabouts. I went to Harvey Mudd College in 1962 and studied there for four years. I studied engineering. It was such a small school that they didn't specialize in different kinds of engineering. They gave a general engineering degree and that is what I got. When I graduated from Harvey Mudd I went to Stanford University on an NSF

Fellowship. Stanford University was just starting their computer science department at that time, but I wasn't sure that this computer business was going to pan out. I thought it would be a more conservative, safer choice to stay with the electrical engineering department, which was a well-established department with a very strong reputation. So I went to Stanford graduate school in Electrical Engineering, with a minor in computer science. While I was in school, I worked at several companies during the summers, including Hewlett Packard, General Electric, and Lockheed Research Lab in Palo Alto. I interviewed at a number of companies when I graduated and the company that seemed to be closest to what I wanted to do was IBM Research. I liked the idea that my work would be directed toward a product and that I would be doing advanced computer science research that eventually would have a commercial application. So I joined IBM Research in Yorktown Heights, New York. I moved from California to New York in February, having lived in California all my life. That is something I don't recommend doing.

Frana: There was no IBM Research facility at all in the area at that point?

Chamberlin: There was actually a research lab here in California.

Frana: Where was that?

Chamberlin: It was located on the main IBM site down on Cottle Road, where IBM designs and manufactures disk drives. Research occupied just one building out of many on the main plant site in those days. Actually, I skipped something here. My first introduction to IBM Research was as a summer student. I was offered a summer internship at Yorktown Heights when I was a Stanford graduate student and I drove across the country. I had just got married so my wife and I made kind of a honeymoon out of it. We drove our '62 Chevy from California to New York and spent the summer at Yorktown Heights. And I really liked it there a lot. I met a lot of interesting people and did some very challenging work and I thought it would be a great place to work. And so that was really what influenced me when I graduated to go back there full time.

Frana: So you'd had a chance to experience it there and you were pleasantly surprised. That's interesting because there do seem to be these cultural differences between Yorktown Heights and San Jose that cropped up when I was reading through the transcript of that reunion meeting you had a few years ago. There seemed to be a different ethic of work there between the people that were from the East Coast and the people from the West Coast. But you didn't notice that so much?

Chamberlin: No, I don't think I'd describe it as a different work ethic. In fact, everywhere that I've visited IBM in the world it is kind of a homogeneous culture. You can go to IBM in Germany or Japan and you find people doing pretty much the same kinds of things with pretty much the same kind of cultural framework. In the early days of my career at IBM back in the 70's, I think the company had more of a flavor of internal competition. It was very prominent in the marketplace in those days and you would often find that in some particular area there might be several projects working in internal competition. Then when a product would eventually be released it might be based on one of those internal projects or possibly on some combination of them. That was part of the corporate culture. I think it is less so nowadays. I have found that

there is more of a spirit of cooperation in the company nowadays, possibly because we have more competition on the outside to worry about.

Frana: Was there more duplication of activity in those days too, where different facilities were overlapping in their responsibilities and the things they were working on that caused that kind of competition?

Chamberlin: Well, that would sometimes happen. Database software is a case in point. In the early seventies it developed that there were database activities in the research division in several different locations. And the division decided in 1973 to consolidate these different projects in one place, to bring resources together and achieve critical mass to mount a major effort to investigate Ted Codd's relational database idea. That decision drew together Codd and the people who were working with him here in San Jose, the group that I was in at Yorktown, and some people from other locations as well. Raymond Lorie, for example, came from the Cambridge Scientific Center to join the project.

Frana: You know, compared to some other disciplines, database really seems tight, like everyone who has done significant research really does know one another. Even outside of IBM there seems to be all of these links between people. I want to back up for a minute and just ask you, were your parents engineers? Were they scientists or not? Were they IBMers?

Chamberlin: No, my father was a high school teacher and a principal later in his career. He was an English teacher. My mother was a homemaker. She never worked outside the home.

Frana: And your advisor up at Stanford? Did you have a particular person that was guiding you?

Chamberlin: My thesis advisor was Ed McCluskey, since I was in the electrical engineering department. McCluskey was a hardware guy. He had done a lot of work on minimization of logic circuits and reliable hardware design and things like that. My thesis was actually a design for a parallel machine. Nowadays it would be called a dataflow machine, although we didn't use that term back in those days. I was influenced a lot in my thesis work by McCluskey and also by Bill McKeeman, who was a young professor at Stanford in those days.

Frana: Now I take it that you didn't specialize in database design, obviously you couldn't, but where does that come from? Where does this discipline of database come from?

Chamberlin: That was an accident as far as I was concerned. I went to Yorktown to work on an operating system project called System A. That was the work that I had been involved in as a summer student. I was interested in software and operating systems and we were building a very advanced system, called a time-sharing system in those days. In the early days computers were very expensive and people were relatively cheap, so you had one computer dedicated to many people. Now it is kind of the opposite. Anyway, in those days, I was working on a time-sharing system that could keep a lot of people busy at once. It was an interesting research project—we were working on virtual memory, which was a new idea. That is what I went to Yorktown to do. As it turned out, this was one of those cases where there was some redundancy in the company, which wound up getting consolidated. The operating system project at Yorktown was

consolidated with some product development work in Poughkeepsie, New York. The people in research who were working on the operating system project got an opportunity to go to Poughkeepsie if they wanted to, but I didn't want to do that. So I stayed in Yorktown and looked for something else to do. There was a charismatic manager in my area named Leonard Liu. Leonard took the initiative to start something new for the people involved in this operating system project who were left behind when the project went away. Leonard had been doing some reading and became interested in the database work that was going on in San Jose and other places. He thought that this was a fruitful area for research and he began to get some research work started in Yorktown on a daily basis. The work had several parts. There was database design work and there was also work on query languages. I became more interested in the query language part.

Frana: So database comes from demand paging memory allocation strategy mechanisms?

Chamberlin: Well, no, not so much. Those are more operating-system kinds of issues.

Frana: File management issues?

Chamberlin: File management is a lot closer than demand paging. Demand paging is what an operating system does in order to give the illusion of a large amount of main memory by paging part of the memory off to disk.

Frana: And it was used in the first 370 system, is that right?

Chamberlin: Yes. Actually, there was a 360 that had demand paging. It was called Model 67. And then the 370 line came along and many different 370 models had demand paging. But the term database has more to do with what we call persistence. That is information that is stored in the system indefinitely. Demand paging is for transient information that is just part of memory during a particular computation.

Frana: It can be called back quickly if necessary?

Chamberlin: Yes. But a database is a store of persistent information and it is also a discipline for accessing that information by multiple users who may want to retrieve and update the information, so you may get into issues where multiple users are trying to update the same piece of information at the same time, or trying to update several related pieces of information using transactions. A user may want to bundle several updates together into a transaction, so that if one update is effective they will all be effective–then the transaction can be viewed as a unit. That gets into some very interesting problems about how to put locks on the data so access is granted to one user at a time. You want to give the illusion that users are sharing the information, but to make sure that they don't share it in a way that allows them to interfere with each other.

Frana: That's very nicely put. I've never heard it put quite so clearly. So time-sharing really does drive some of the development here in database management?

Chamberlin: In fact, concurrent access to shared data is central to the whole discipline of database management. Also very important is the idea of recovery. For example, suppose you are a company whose livelihood depends on stored information, such as a bank or a stockbroker, and you are processing thousands of transactions per second from customers all over the world. Things do fail from time to time. There could be a rolling blackout, your power could fail, and your machine could go down right in the middle of your working day. Well, you need a system that makes sure that if there is a failure, the system fails in a clean state so there isn't somebody whose transaction is half done. And then when things come back up, you need to recover the state where everything was at the point of failure. So a recovery system usually has several parts. There are image copies where once a day or on whatever frequency you choose, you dump the whole database onto some offline medium like a tape. And then during normal operation there is a log, in which every action gets recorded not only in the database, but also in some other medium from which it can be recovered if the database gets lost. It is possible that you could have a disk crash and your database could be wiped out. By using a combination of a recent image copy and the log, you can reconstruct the state of the database. So recovery is another important part of a database system.

Frana: And when you were at Yorktown, is that when you heard about Ted Codd's work?

Chamberlin: Yes, it is. Leonard Liu was trying to get some new research areas going and so he would assign us to read papers and report on them. We studied the work that Ted was doing with the relational model of data. We also studied what was then considered the mainstream of the database industry, which was the CODASYL work. Ted's work was mainly of academic interest I would say. It was considered to be a little bit out of the mainstream, somewhat mathematical and theoretical in its flavor. But the CODASYL work was more of an industry standard, developed by a consortium of companies that were working together to design a standard database language.

Frana: We have a large number of the CODASYL papers at CBI.

Chamberlin: Yes. There was a guy named Charles Bachman who worked at Honeywell, I believe. He had developed a system called the Integrated Data Store. The CODASYL standard was based loosely on this Integrated Data Store or IDS. It wasn't exactly the same but it was very similar. The idea was that you had a network of records and these records could point at each other. And the way that you would use this network of stored data was by writing a program that would navigate through the network following the pointers or links from one record to another.

Frana: Was this a hierarchical network, or a distributed network?

Chamberlin: The CODASYL proposal was not hierarchical—it was more general than that. Different manufacturers had different data models in those days. They mostly shared the "navigational" property, meaning that they represented information by records that were connected together by links or pointers. IBM's navigational database system at that time was called IMS, which stood for Information Management System. IMS was a hierarchical system, which basically just meant that the records were connected together in a way that formed a

hierarchy. A query would search for information by starting at the root of the hierarchy and then moving to the children and to their children and so on. CODASYL was based on a network data model. It was a little bit more general than the hierarchic data model because it didn't have the constraint that data had to be organized into a hierarchy—the records could be organized in whatever way you like.

Frana: So it really was distributed? Every point was equal in value and each pointer could link to any potential piece of data?

Chamberlin: That's true of the CODASYL model. I learned to write queries in the CODASYL language, and one day we had a symposium at Yorktown Heights. Ted Codd came out to visit us from San Jose. It was the first time I had met him. He gave a speech and he said, 'Sure, you can express a question by writing a navigational plan. But if you think about it, the navigational plan isn't really the essence of what you are trying to accomplish. You are trying to find the answer to some question, and you are expressing the question as a plan for navigating. Wouldn't it be nice if you could express the question at a higher level and let the system figure out how to do the navigation?' That was Codd's basic message. He said, 'If you raise the level of the language that you use to ask questions to a higher and less procedural level, then your question becomes independent of the plan. As circumstances change, the computer might change the plan. You'd never even know the difference. You are just thinking about your question, you are not thinking about exactly what the computer has to do to find the answer.' It is kind of like what had been done years before with high level programming languages. You know, the first programming languages operated directly on the computer hardware so you were aware of the registers and the arithmetic unit…

Frana: … machine language, yes.

Chamberlin: …and all of the operation codes that the machine could process. You used all of these things directly. If you wanted to add two numbers together you had to load them into registers first and so on. All this changed when the first high-level languages came along. John Backus worked on FORTRAN at IBM in New York City. He said, 'Why don't we just write an algebraic formula here? The person who is trying to evaluate this formula doesn't care about registers. Let the machine figure out how to do it.' Well, at first people were horrified by that idea because they were really good at allocating registers. And they said, 'How can any machine do this job as well as I can? If we let machines do this automatically they will do a bad job. It will be inefficient.' Well, the same arguments were made about relational database systems. Some queries involved quite complex plans where you have to decide which index to use, and which records to retrieve, and what order to do joins in, and things like that. There was a lot of skepticism that computers could do a good job of automatically generating these plans because it seemed to involve a lot of intelligence. Basically the same debate played itself out. People didn't believe that it could be done until somebody did it. And that's really what the System R project was about—it was about proving that this kind of automation was in fact possible, and not only possible but efficient.

Frana: So was Codd his own best popularizer? I've heard this both ways. Some people say 'Oh, Codd, he was so mathematical that it was difficult to comprehend what he was talking about.' It

sounds to me like you immediately realized what he was talking about and that he presented it in a very clear fashion. I mean it seems so obvious to me having these relational databases since then that it makes sense, but is it that the network model had such a hold on people, these other models had such a hold on people, that they couldn't really get into Codd's referential point of view? Or is it something else? Is it the mathematics? Did it take time to get the message across to people?

Chamberlin: Partly, I think, people had a large investment in the status quo. They had systems implemented and large databases that were operational and that were based on navigational models, and whenever people have a large investment in something they are not going to turn a sharp corner. It is going to take a while for new ideas to gain acceptance. I think that Ted was an effective person in explaining his ideas to academics and to people who were working in the field. I think the idea that you could write queries using a mathematical language was pretty easily understood by experts, but it wasn't very easily understood by the next level of people.

Frana: The people who were actually using and maintaining these databases?

Chamberlin: Yes, that's right. The ideas didn't have immediate appeal to practitioners, I think, largely because Ted couched them in mathematical symbolism and terminology. In his original query languages he used mathematical notation, like universal quantifiers and existential quantifiers, and he used Greek letters a lot. Things like that just give the appearance of something being very esoteric and difficult to deal with. Whereas, actually, what he was trying to do was to make queries easier to write, not harder. So I think the development of a language based on English keywords, which you could type on your keyboard and which you could read and understand intuitively, was a breakthrough that made it much easier for people to understand the underlying simplicity of Ted's idea. It didn't really make the ideas any more simple; it just made them look simple.

Frana: Sure. Right. Now did you move out here before or after System A was discontinued?

Chamberlin: After. When System A was discontinued, my group needed to find a new project. We got involved in databases. We were one of several database projects that were consolidated in 1973.

Frana: And how many of you moved down here, by your reckoning?

Chamberlin: Well, we came out in two waves, I think. I came out here from Yorktown Heights with Leonard Liu, Frank King, Ray Boyce, and Vera Watson. Five people. The following year we were followed by Mike Blasgen and Franco Putzolu.

Frana: Mike is now at Sony, is that right?

Chamberlin: Mike is retired now. He was at IBM for a long time, then he was at Sony for a while, and he is now retired.

Frana: And where else did people come from? Where else did they migrate from other than Yorktown Heights?

Chamberlin: Cambridge Scientific Center, I think, was the only other IBM location that contributed personnel to System R. That was Raymond Lorie.

Frana: And so the team that was assembled here was how many, thirty people?

Chamberlin: Oh no, not nearly that big. Initially I'd say twelve. It might have grown a little bit. It had a life cycle like any project. The peak was less than twenty I think.

Frana: Okay, all right. Now when you came out here Ray Boyce was here?

Chamberlin: No. Ray Boyce was with me at Yorktown Heights. We transferred to California at the same time.

Frana: And you worked with him very closely? In that reminiscence you talk about almost sharing a desk. Was it that way when you were working on SQUARE?

Chamberlin: We didn't literally share a desk or an office. We had separate offices. But we were close collaborators. That's something that is fairly common in IBM. There have been times when I have worked very closely with somebody. I enjoy that kind of work. It kind of amplifies your own ideas to bounce them off another person.

Frana: The reason I ask this is that I was down at UT Austin two months ago, and there is an IBMer, an old IBMer working down there, Glenn Henry, who managed development on System/38. And he said that there is an optimal size for research groups and it's actually much smaller than you might imagine. Do you have, in your mind, what the optimal size is for this kind of fundamental research in computer science? At what point does it get so large that it is difficult to do really cutting-edge work?

Chamberlin: I think it depends to some extent on the domain of the work. I know that we have a theory group here where there are theoreticians who work all by themselves or in very small groups like pairs, proving theorems and studying the complexity of algorithms, and things like that. For systems work you need more resources than that. To build a system with hundreds of thousands of lines of code, you need a significant sustained effort over a period of time by a group of people. I think that System R was just about the optimum size from my experience. Twelve people are enough that you can specialize to some extent in different parts of the project. You have a lot of background, people with different expertise that they can bring to bear. But it is not so large that you run into communication problems or bureaucracy. So I think on the order of twelve very good people with mutual respect and a spirit of cooperation is about the right size for a systems research project.

Frana: Now has that changed over time? Has that number fluctuated, not specifically for your project, but I imagine IBM in the '60's as this vast army of people wearing blue suits and it's the aggregate that makes IBM so powerful. But then, by the late '70's and '80's, there is this sense

that IBM is about flexibility, about really small working groups, and people even working as individuals and that is how great things get started. Is that fair to say? Or is that just sort of the model people have, and the reality is something different?

Chamberlin: Well, first let me say that I've worked for IBM for thirty-one years and I've never worn a blue suit once.

[laughter]

Chamberlin: And I also think there are different styles of working in different parts of the company. Research tends to be a more individualistic place. Now if you are building OS/360, you need a cast of thousands to sustain a project like that. You have to document it and you have to maintain it. There are lots of jobs that just have to be done and you need a large organization to make them possible. We were trying to do something that had not been done before. We were trying to compile a high-level query language into automatic plans for database access. And it wasn't clear that could even be done. So in a project like that it is really not helpful to have five hundred people on the first day. You wouldn't know what to tell them to do. What you need are a few very thoughtful and expert people who can get together and bounce ideas off each other and try things out to see if they work and be flexible enough to abandon them if they don't. And you can't necessarily do that kind of work on a schedule. You can't say, 'We are going to release it on February 15th.' In the product world you have to be able to do that. You have to be able to build business plans and educate your sales force and so on. But we didn't have anything to sell. We were doing research. I think the image of large legions of people all working together toward a common goal is probably an accurate one in a mature product line. But in research there is a different style of work. In a system research project, I think a group of twelve is probably typical. And I don't think that that has changed particularly over time. I think the same is true of other groundbreaking research projects, such as the group at Xerox PARC that built the Alto. If you talk to those people, my guess would be that you would hear somewhat the same story.

Frana: You may not want to talk about this too much, but has there been more pressure in recent years to produce products? I am thinking here of Bell Labs' Lucent Technologies or 'Let's put this research on one track and get right to the product as quickly as possible.' Has that materialized here?

Chamberlin: I wouldn't call it pressure. I think that there is more product-related work going on here now than there was twenty years ago. But I think the reason for that is that our ideas have been very successful and research people tend to be motivated. They are turned on by seeing their ideas put to use. So now that IBM has some successful relational database products that were based on the outcome of our early work, naturally people want to get involved in these products and add new features to them and plan their future evolution. People do that because they want to. There isn't anybody making them.

Frana: Thanks for taking that excursion. Getting back to your work at that moment when you moved to San Jose, were you thinking then, 'I want to build a high-level query language that is useful to lots of people. I'll take a relational model, I'll develop a query language for it, and it will be very user friendly?' Is that what you were thinking then?

Chamberlin: Yes. That is what Ray and I both wanted to do. We had fooled around with this language that you alluded to called SQUARE. SQUARE was an acronym that stood for Specifying Queries As Relational Expressions. That was our first attempt to make Ted Codd's ideas a little more user friendly, or to build a language around them that might be a little bit easier for people to use. Codd's initial languages, in our view, had some shortcomings. One was that they were couched in mathematical notation and you couldn't type them on a keyboard. Another was that he only considered how to ask questions and he didn't have any features for modifying or updating data. Of course, updates are an important part of the problem. If you are running a bank you need people to make deposits. Any real database application such as airline reservations or banking is going to involve modification of data. So we wanted that to be part of the language. And there were a few other things that were missing from the original languages, such as the operation we now call "group by". There wasn't any grouping operator in the relational algebra or calculus as Codd envisioned it. Ray and I had been working on query languages in Yorktown. We had a little game that we played called the query game, where one of us would dream up some question and see if the other one could find a way to express that question in a computer language. When we came to San Jose, that became our part of the project. We worked together on developing a syntax based on English keywords. There was a woman here in San Jose named Phyllis Reisner. She was a cognitive psychologist and she liked to try to measure how easy to use things were. That's kind of an amorphous thing, and hard to measure. Phyllis liked to do human factors experiments where she would get a bunch of college students together and try to teach them something, and she would measure how well they could learn it, how long it took to teach them, what kinds of problems they had and so on.

Frana: Was Phyllis an IBMer?

Chamberlin: Yes, she was here at IBM San Jose when I arrived. She actually did a comparative study using some student subjects at San Jose State University. She tried to teach them these two languages, SQUARE and SEQUEL, to see which one was easier to learn.

Frana: Were these engineering students, computer science students, or just regular undergraduates?

Chamberlin: She actually ran two batches of subjects through her experiments. One of them consisted of undergraduates who had had no exposure to computers at all and the other consisted of undergraduates who had had some programming experience. Possibly they had taken a programming language course, although they were not necessarily majoring in that area. Phyllis had some interesting results.

Frana: And she really did help you modify the interface, the way that people accessed and used SQUARE at that point? At what point does it get renamed SEQUEL or SQL?

Chamberlin: Well, SQUARE was the work that Ray and I did in Yorktown. It still had some mathematical notations in it. It wasn't based on English words. SEQUEL was the successor to SQUARE. You might say it was the 'Sequel' to SQUARE. And again, it was an acronym, for Structured English Query Language. It was our first attempt to base the language exclusively on

English keywords so that you could read a query from an intuitive understanding of the English words. A query might say something like, ‘Select salary from employee where manager equals John Smith.’ From the English words like ‘select,’ ‘from’ and ‘where’, you can get a pretty good idea for what the query is about.

Frana: Yes, and at that point, object orientation was not something that people were thinking about?

Chamberlin: Object-oriented systems didn't come along until several years after System R.

Frana: And when did System R get that name? You’ve got System A, is there some rhyme or reason for the picking of the letter?

Chamberlin: Well, Frank King was the manager of the project. And one day Frank said, ‘We cannot have a project without a name. You guys have to come up with a name for this project.’

Frana: And you’d been working for several years?

Chamberlin: Oh, not several years. We’d been working for probably six months. And Frank said, ‘We are going to write some papers about this, we are going to get some publicity for it, we are going to talk about it in internal meetings in the company, and it has to have a name. So we thought about that for a long time. We finally decided that we’d call it System R. For relational. Franco Putzolu always claimed that R stood for Rufus, which was the name of his dog.

[laughter]

Frana: That’s a great aside! Now is the relational model best suited for business applications or was that just the motivating factor because this was IBM and business applications often seem to dominate all activity?

Chamberlin: I wouldn’t necessarily say that the relational model is better suited for business applications than for, say, scientific or engineering applications. In fact, the System R project built a prototype in order to get feedback from some IBM customers. We installed this experimental prototype for free in three customer locations and we had what we called joint studies between the customers and the people at IBM Research who were trying to evaluate the prototype. The studies continued for a period of two years. We learned a lot by getting feedback from real people using the system in real applications. Two of those customers—you might even say all three of them—were scientific rather than business applications. The first one was Pratt and Whitney, the jet engine company in Hartford, Connecticut. Although that’s an engineering business, they actually used the system for an inventory control application. The second joint study that we got operational was at the Upjohn Drug Company in Kalamazoo, Michigan. They used the system for keeping records of clinical trials on experimental drugs.

Frana: To keep track of the various inventory numbers.

Chamberlin: They were gathering information for applications to the FDA to get approval for new drugs. And the third joint study customer was Boeing Aircraft Company in Seattle, in their engineering division—I believe they were designing the 757 back in those days.

Frana: And what did you learn in those three joint studies? Anything that you didn't realize when you set out to implement System R?

Chamberlin: I don't think I'm going to be able to give you a short answer to that. We learned a lot.

Frana: It's in the papers.

Chamberlin: Yes, it's in the published papers.

Frana: Yes, I certainly don't want to go over what's been so well done in the papers. What's the relationship between UC Berkeley and IBM Research out here?

Chamberlin: Jim Gray and Mike Blasgen were both engineering graduates from Berkeley who worked on System R. The people at Berkeley, notably Mike Stonebraker and Gene Wong, had read Ted Codd's papers and understood their potential, and were trying to do exactly the same thing that we were. In their Ingres project, they were exploring the idea of building an industrial strength implementation of Ted's ideas. They developed their own query language, which was called Quel. They built their own optimizer, which translated Quel into lower level access paths, and they explored many of the same issues of logging, transaction control, and concurrency that we studied in System R. I guess you could say that the two research projects were working in very closely related areas with some exchange of personnel and occasional exchange of visits. We'd give a seminar at Berkeley and somebody from Berkeley would give a seminar down here. There was a certain degree of rivalry between the two projects, as you might expect.

Frana: It sounds like there was. Would Ted's paper have been published today? He published it out there for the world to see. Berkeley saw it and understood it and picked up on it. Not that Berkeley would be in direct competition with IBM, but would his paper have remained proprietary longer today? How do you protect these ideas at a time when there is no software patenting?

Chamberlin: I think IBM Research has had a tradition throughout its history of publishing its results in the open literature, and still does that even to this day. I think that is a healthy thing. If IBM had tried to keep Codd's relational ideas internal and not share them with anybody, I don't think they would have been nearly as successful. Remember, IBM had its own commercial database product in the '70's, called IMS. The work on IMS, of course, was considered proprietary. The work on the relational model was considered more exploratory in nature. For that reason we were free to publish it in the open literature. We wrote papers, and the folks at Berkeley wrote papers. We benefited from collaboration and from bouncing these ideas off each other. And the relational model benefited from the fact that there were a lot of people working on it and it didn't belong to anybody exclusively. Look what happened to Oracle and Larry Ellison. Larry read some of the research papers from the System R project and built a very successful

company around them, and legitimately so. He made his own contribution to the implementation of the relational ideas and he was very successful. So I think the fact that many different people in many locations were working on these ideas, discussing them at academic conferences and sharing their ideas, was a big part of what made the relational model successful.

Frana: It actually made IBM Research stronger to have external colleagues and to have an audience out there for the ideas?

Chamberlin: Yes. I also think it caused IBM to take our work more seriously, to realize that there were people outside the company who were following it very closely.

Frana: Maybe this isn't the best moment, but while we are on it, do you have an opinion on software patenting? Do you feel strongly one way or another about it? Don Knuth, whom I am interviewing next month said that, well, 'Software and mathematics come from God. You can't patent God.' Do you feel strongly about that one way or another, or is it just another way to protect what people have done; the work that they have produced?

Chamberlin: That is a really complicated question.

Frana: Yes, it really is.

Chamberlin: I don't have a really strong opinion on one side or another on the software patent issue. I can argue various positions with respect to software patents.

Frana: Do you have a number of patents yourself?

Chamberlin: I don't actually hold any software patents. I think that it comes down to how software patents are administered. If they are used in a way that suppresses new ideas and competition, then I think it is harmful. Of course, that is not the original intention of the patent program. The purpose of patents is to reward people for their innovations and to encourage innovation. I think it is a fine line to administer software patents in a way that encourages innovation rather than suppressing it. That's a hard thing to do because lawyers run the system and lawyers certainly aren't software experts. So I think the system as it currently stands has some limitations, but it also has an underlying good purpose. It's a mixed bag.

Frana: We live with absurdities like a one-click button that someone can claim to have a patent on. But by and large we muddle through and we are actually very effective. Don [Knuth] has said that eventually all this activity is going to move overseas because it just can't be done here in the United States. You don't perceive anything like that?

Chamberlin: I haven't observed any large migration of software projects.

Frana: You aren't moving to IBM India at this moment?

Chamberlin: Well, IBM does have a research lab in India and a lot of software is written in India. But up to now I haven't observed a major trend against the domination of the software industry by this country.

Frana: Thanks for answering that question. I know that you are more of a technical expert as you perceive it, but you are actually a very good historian. I am pleasantly—well, not surprised, given what else you've written. Maybe I should ask you, why is the history of System R so well documented compared to some other really important projects at IBM that haven't been? Is it because there have been a few people who have organized and tried to lay out the history of System R?

Chamberlin: That's part of it. The fact that Jim Gray, and Mike Blasgen and Paul McJones…

Frana: And yourself.

Chamberlin: …organized a reunion at Asilomar and recorded it and put it on the Web I think is part of the answer. [See http://www.mcjones.org/System_R] Another part of the answer is that I think the people who worked on System R really had a good time. They were all young people at the beginning of their careers. They saw their ideas become widely accepted and form the basis for what is now nearly a $10 billion a year industry. That is something that you have to look back on with a certain nostalgia. It was a very exciting time in our careers.

Frana: You are taking pride in meaningful work.

Chamberlin: Yes.

Frana: That's great. Do you have plans for a future reunion?

Chamberlin: I haven't heard any so far.

Frana: Let me know if that does materialize. How did System R then get transferred outside of the research laboratory? When I was down in Austin talking to Glenn Henry he said that System 38 was the first small system that incorporated relational database technology. But he said he'd never heard about any of you guys and he hadn't read any of your papers. I was dubious about that. How could he have learned about this if not from reading something or hearing about it second or third hand? I mean, how are ideas communicated out of here? How were they in the mid-seventies?

Chamberlin: Well, I'll tell you how it started. Once again, it came about partly by accidental circumstances. There was a group in Endicott, New York, whose project had been completed or terminated for some reason and they were looking for work. The lab at Endicott owned a mid-range operating system called DOS/VSE that needed a database system. At just about the time that we were completing our joint studies with the three experimental customers and trying to get IBM interested in releasing a commercial product, these guys were looking for a database system, and it seemed like a good match. So we actually exchanged some personnel in both directions. We sent some people—Bob Yost was one—from the System R project to Endicott to

work with these guys and teach them about our code so they could take it over and adapt it to their environment. And they sent a couple of people to San Jose too—Art Gilbert and Tom Sigielski—they came out to San Jose to become steeped in our culture and to take over ownership of the code. The first commercial system that was based on System R technology was called SQL Data System. We changed the original name "SEQUEL" to SQL because we got a letter from somebody's lawyer that said the name "SEQUEL" belonged to them. We shortened it to SQL, for Structured Query Language, and the product was known as SQL/DS. The initial product was released in about 1979 or 1980, out of the IBM lab at Endicott. Now, as you know, IBM has many platforms and it is a very large distributed company. This particular system lived in Endicott, but right here at the Santa Teresa Lab in California, the home of IMS, they were looking for a new generation database system for the mainframe. They took a hard look at System R and adapted it for that use, resulting in the DB2 product, which was released about 1981. The IBM Personal Computer was also released at about that time, and IBM thought it ought to have a database system. That mission went to Austin, Texas. For a while we collaborated with the guys in Austin to move an implementation of relational ideas into the PC world. Before long, we noticed that Oracle was being very successful with a UNIX based relational database. IBM wanted a piece of that pie so we developed a version of DB2 for the UNIX platform. That was done in Toronto, Canada. Over a period of time, the ideas from System R were disseminated to a lot of different locations in IBM.

There was a woman here at Almaden named Pat Selinger. You may have heard of Pat. She developed the world's first cost-based query optimizer, published a paper about it, and became quite well known for that work. Her paper is still referenced in university courses. She became an IBM Fellow and was elected to the National Academy of Engineering. She was one of the central figures in the System R work. She also played a central role in disseminating results from the research lab here to the product divisions of IBM. She was the founding manager of something called the Database Technology Institute, DBTI. DBTI was an organization based here at Almaden and staffed with rotating personnel from many different locations. People were often assigned to DBTI for a year or two from some development organization. They would come here to learn about relational database technology and then they would go back where they came from and become the local expert, transferring ideas into a product line. That work is still going on. DBTI still exists.

Frana: Is Pat still around?

Chamberlin: Sure, Pat is an IBM Fellow. She is no longer the head of the DBTI. She is now at the Santa Teresa Lab, which recently changed its name to the IBM Silicon Valley Lab, and she has some new responsibilities. For a while her job was to make sure that SQL is implemented consistently on all the IBM platforms, which is a pretty big job and harder than you might think. Now she is looking at what is called information integration, which is trying to put a capability into DB2 for processing XML and other kinds of data.

Frana: Would she be a very good person to talk to with regards to this project? Does she have a long history with IBM?

Chamberlin: Sure. I was Pat's first manager in 1974 or thereabouts. She came to us with a Ph.D. in applied mathematics from Harvard. As I said, she built the first cost-based query optimizer, and that was a key part of proving the commercial viability of the relational data model. So she has been a central figure right from the start.

Frana: I'll file her name away and see what happens. And if there are other people as we go along that you think are crucially important to do oral histories with I am happy to take those suggestions.

Chamberlin: Well, you might think of Jim Gray.

Frana: Definitely Jim.

Chamberlin: Jim is a Turing award winner. And, of course, he has worked at IBM, then at Tandem, then at DEC, and then at Microsoft. So he has seen the whole industry.

Frana: Yes, he seems to have a very nice birds-eye view of what's going on. And I know that the *New York Times* relies on his wisdom from time to time. He is up in San Francisco, right?

Chamberlin: That's right, yes...

Frana: Moshe Zloof, what is the relationship there with QBE and Moshe Zloof? He is on the other side of the country, right?

Chamberlin: I am not sure where Moshe is right now. He was working here in California for Hewlett Packard for quite a while. He may have retired now. I am not sure. Moshe developed what would now be called a graphical user interface. We didn't use that term in those days, but that is what it was. It was an interface for composing queries and updates for a relational database. It was a good idea. It was done at Yorktown at about the same time as System R. This was after the first wave of people had come out to San Jose to work on System R. As I said, in those days there was a lot of redundancy in IBM. You'd find the same kind of work being done in various different places in the company. Some relational query work still continued in Yorktown independently of the System R work, and resulted in this graphical user interface called Query By Example. When the time came to commercialize these ideas into mainstream IBM products, Query By Example became one of the user interfaces to DB2. In other words, it was one of the strands of research that fed into the product line.

Frana: Do you want to say anything about Eagle?

Chamberlin: Eagle was just an internal code name for a database development project at one point at the Santa Teresa Lab. Before a product is announced, as you can probably imagine, it has some internal code names and these change from time to time. Eagle was one of the code names that was given for a certain period of time to the work at Santa Teresa to commercialize the relational database ideas.

Frana: And I just want to lay out the timeline right here. Is there a product announcement for System R? Did you go through the same kinds of review processes that some of these large and small systems go through where there is a committee to try to determine six or nine months in advance whether it is really going to be accomplished and ready for customers?

Chamberlin: System R by itself was never a product.

Frana: It was never a separate thing?

Chamberlin: No. It was strictly a research prototype.

Frana: It was always integrated?

Chamberlin: It was intended to prove the concept that it was possible to implement a relational query language efficiently. The research division of IBM doesn't typically produce products *per se*. We produce technology. We transfer the technology into products using mechanisms like DBTI. You might say that System R was the forerunner, first of SQL/DS, then of DB2, and eventually of the whole line of relational database products. But it was never a product in its own right.

Frana: So it gets turned over to the Institute and then there's a transition process that goes on there?

Chamberlin: Yes. Sometimes some of the code will actually survive and be adapted to different platforms. But the process of writing the user level documentation and setting the whole thing up for large-scale production and maintenance is beyond the scope of what we can do here in research.

Frana: Okay. That's really important. Apart from experimental applications, do you get feedback from actual customers? Does that come back directly to you when you are doing research on future systems, so that you know what the experience has been out there? Does the Institute tell you that or give you that information?

Chamberlin: Yes, the DBTI does some of that. And also people from research are often called in to consult on some critical situation that may be going on somewhere. Somebody is trying a new application and is not getting the performance that they expected, or needs some advice on database design, or something like that—we will often be called in on a situation like that.

Frana: Now in the long view, has the distance narrowed between the end user and fundamental research? I mean clearly that was the objective in part of your work here, and still is, but are there initiatives to close the gap, so to speak, between the fundamental research and the end user?

Chamberlin: Yes. I think that as these ideas have become more successful, naturally there are more people using them, and getting into different kinds of trouble with them and needing our help. So I think over time there has been an increasing involvement of that kind.

Frana: Did unbundling have any effect on your work at all? We had a big conference out here in Palo Alto last year with a bunch of software pioneers and the subject was IBM's decision to unbundle. And it was a very fruitful discussion. Did it impact you at all?

Chamberlin: I don't think it did. From the very beginning we were a software project and we viewed software as our end product. We viewed it as a commercial product in its own right, not as an adjunct to any particular computer. And, of course, the technology that we developed has been used on many different platforms. It was never thought of as being directed toward a particular piece of hardware.

Frana: What about knowledge acquisition and knowledge management problems? Now you've got all this data and there are all sorts of neat things that you can do with it, but there is so much of it. Some of it is irrelevant and some of it is very relevant. You've got to sort out the needle from the haystack. Did you become interested in that in the mid-seventies? Did you recognize that there was this knowledge acquisition bottleneck developing?

Chamberlin: Well, I certainly recognize that what you describe is a big problem and it is getting bigger all the time. That hasn't been my particular area of specialization within the database field. But there is work going on here at Almaden in that area.

Frana: Data security, is that something that is of interest to you?

Chamberlin: Sure, data security is another important aspect of the problem. In fact, in System R one of our earliest publications was about an authorization system for granting privileges to access particular pieces of data, and allowing the recipient to make further grants and so on. There were some interesting kinds of theoretical problems involved in that area.

Frana: What about dealing with heterogeneous sources of data? I was recently talking to Gio Wiederhold. That's of especial importance to him. You've got all these different data sources—do you standardize them right away before you even accumulate the data? Do you do it at the end somehow? Have you been interested in blending information?

Chamberlin: Well, there again that hasn't been a major focus of my work personally. I think the System R people tended to view data as being formatted in relations, or tables if you like, and not to particularly focus on where the information came from initially, how it got to the table. I think there's a lot of interesting work going on nowadays around using XML to represent heterogeneous data. XML is a new markup language for what you might call semi-structured data that carries its own description along with it.

Frana: Our software history dictionary eventually is going to be rendered in XML. Which makes sense.

Chamberlin: Yes. Traditional databases tend to be oriented toward data that is very well structured and very repetitive. For example, you might have many bank accounts, but all the bank accounts basically have the same kinds of information. XML, on the other hand, came out of the heritage of the publishing industry and is a way of representing information that is not

homogeneous. You can represent books in XML, but not all books have the same things in them. If you look at several different XML documents, rather than each one of them being uniform in its structure like a bank account, each one of them will be different. In a relational system, the metadata—the data that describes the data—can be factored out into a separate catalog. The data doesn't have to carry the metadata along with it because it has been factored out, and this works because the data is very repetitive. But in an XML document, the data carries the metadata along with it in the form of tags, and the metadata and the data are all mixed together. The question of how to query data in that format is the major focus of my work right now.

Frana: And how long have you been struggling with that problem?

Chamberlin: Well, I have been representing IBM on the W3C working group for XML query since the group was formed late in 1999. So we are now almost two years into this work. The working group has representatives from many different companies. There are probably thirty-five members in the group. We have been studying the problem of how to query XML data. For me, it's like going back to my roots. I am having a lot of fun designing a new language from the ground up, which I haven't had the opportunity to do for a number of years. It is very interesting work.

Frana: But content management, when in your mind did this become an important problem? Certainly before 1999? When did you see that we need to think about metadata? At the Charles Babbage Institute, we use Encoded Archival Description Language, EAD. We have a huge archive and we are trying to figure out a way to structure this—you know, librarians love to structure things. The Web, in particular, seems so unstructured to them. To give you a little more history, the University of Minnesota is where Internet Gopher comes from. That's highly structured. The librarians across the country loved Gopher! The Web they have a lot of problems with. They just don't know how to handle that. So, where does this come from? It's got to be older than XML and the Web even, trying to figure out a way to reorder what we've made disordered on purpose.

Chamberlin: Yes, well, I guess ever since Dewey invented his decimal system people have been working on that problem.

[laughter]

Frana: That's true.

Chamberlin: And especially since the explosion of the Web in popularity in the mid-90's, when we began seeing search engines like Google and AltaVista.

Frana: I saw the Google search engine at Stanford. Have you been back to the Gates Building lately? In the basement they've got the original Google hardware encased in sort of a Lego case. It's really quite something.

Chamberlin: Yes. Information discovery is a hard problem, and Google does an astonishingly good job. I've often had the experience of giving an off-the-wall query to Google and it really homes right in. It is remarkable how accurate it is.

Frana: So what you are trying to do now, and this is entirely on point, as I understand it, is to try to bring these two things together, Web search and databases? You mention on your website that you are looking at the intersection of these two technologies.

Chamberlin: Yes. XML, and its predecessor SGML, came from a heritage of the publishing industry. They are markup languages for documents. Originally, markup had to do with controlling the appearance of things, such as calling for a bold font or forcing a page break. At some point people began to discover that it wasn't enough to mark up the parts of a document with instructions about what they should look like. For example, it would be better to label something as a heading instead of labeling it as bold. Then we would have more information—not only about what the thing looks like, because a lot of things might be bold, but that this thing is bold because it is a heading. That gives us information that can be used in a variety of ways. It could be used for indexing, or for generating a table of contents, or for rendering the heading on many different devices, some of which may not have a bold option. For example, some devices might underscore the heading instead of making it bold. The evolution from appearance-based markup toward a more generic markup based on logical roles led to SGML, the Standard Generalized Markup Language. SGML is a predecessor to what is now called XML. So, from the heritage of document processing we get documents marked up with descriptive tags that identify their various parts. And from the database industry we have database systems, which are very good at managing stores of information that are highly structured and homogeneous. I think what is happening now is that the Web has brought these two worlds together. A large part of human knowledge is out there on the Web, organized using markup languages—initially HTML, but increasingly XML. Naturally, if all this information is out there, people are going to want to use it. Today you can use it in a rather unstructured way with search engines like Google. But what if you need to use the information in a more structured way? Suppose you are a car manufacturer and you want to buy headlights. You might want to ask your suppliers to submit proposals for headlights, and you want them all to be submitted in a very well defined format with data that you can rely on to run your business. Well, that requires a higher level of discipline than you are going to find from a search engine like Google. That requires something like a database query language, but the query language has to be well adapted to the information out there on the Web that is not all in tables.

Frana: Yes, right.

Chamberlin: A lot of it is organized using a markup language like XML. So I think the challenge that we are facing now is how to get the discipline, the recoverability, the transaction control, the concurrent access, and all these things that relational databases are very good at, adapted to the less structured world of the Web.

Frana: I see. This is exactly the kind of thing that the University of Minnesota has been struggling with on a much smaller scale in trying to manage the information that it produces and to find a way of rationalizing what's out there in a way that isn't restrictive. It occurred to me last

week that we keep all these metatags hidden. The end user often doesn't see them. And I don't know if I was reading something that Ray Kurzweil wrote- I don't remember exactly where I read this- but maybe the thing to do is to have all the metadata out there. For example, English might be a better language if people just said, "What I am going to say next should be in parentheses." And have a word for that and have it just be right there in the string of characters. Do you think that is the right approach, or should these things be hidden like they are with higher-level languages? Should we see on the screen, Title: IBM Research and then Heading One… and actually see the tags?

Chamberlin: I don't think there is a single human interface that is universally best for all applications. For several years I took kind of a diversion from database work and I worked in document processing and formatting. That is the other half of my background. I worked on an experimental system called Quill. It was one of the projects in IBM Research that did not eventually produce a commercial product. There are a lot of those. Not all of our projects are as successful as System R. What Quill was trying to do was to present two different views of an SGML document so that a user could interact with either one. We called them the logical view and the physical view. In the logical view you'd see all the markup, just like you said. If something were a heading it would be labeled with a tag that said "heading". You could edit the tag if you wanted to. In that way you would be interacting with the logical structure of the document. Or you could switch to another view, and in the other view you'd see what the document looked like. It would be what we call a WYSIWYG display—"what you see is what you get." If all you cared about was what the document looked like, you could edit the physical view, and if you cared more about the underlying logical structure, you could edit the logical view. Whatever change you made would be reflected in the SGML document that was underlying both windows. I always thought that was a pretty promising idea, but as it turns out the Quill project never really went anywhere.

Frana: And why do you think that was? I mean, it's hard to see when you think it is so effective, but that doesn't seem to be the way the world is moving. I was just thinking of the first browsers that I used. You could easily see exactly what was going on behind the curtain and now that just doesn't seem to be interesting to the end user. Should it be?

Chamberlin: Well, there again, I don't think there is a single answer that is best for all people or for all applications. Look at widely used word-processing programs like Microsoft Word or, for more technical documents, Adobe FrameMaker. In a way you could say that these systems implement an interface similar to what I was describing. FrameMaker, for example, is a WYSIWYG System. You can edit the document; you can interact with it on a screen while seeing a representation of its real appearance. But you can also invent tags that represent different roles, different usages within a document, and you can create a style definition that maps those tags onto a physical appearance. So I think that these ideas haven't gone away.

Frana: I wanted to ask you on a different level, are there cautionary tales in your discipline? Stories that get passed around that are really important, that are object lessons to all of you as students and that get passed down? For example the Halloween problem, which is a famous example. Could you maybe explain the Halloween Problem briefly, and are there others like that?

Chamberlin: Sure. I can tell you about the Halloween Problem.

Frana: Was that a real problem? Did it really happen?

Chamberlin: It sure did. It happened one day when Pat Selinger and Morton Astrahan, two of the main people in the System R project, were working together on query optimization. I don't just mean queries in the sense of asking questions, but updates as well. When you are accessing data in a relational database the data looks like a table but there are various access paths that you can use to get to the table. One type of access path is called an index, in which all of the data appears to be ordered by an index key, which is one of the fields of the table. For example, if you have an index by social security number, then it is really easy to find a person that has a particular social security number, and if you access the data using that index all of the entries will appear in order by their Social Security number. So Morton and Pat are working on translating high-level queries and updates into plans that use various access paths including indexes. Well, one example of an update that we were playing around with was giving raises to people. The example said, 'Give a ten percent raise to everybody who earns less than twenty-five thousand dollars.' So Pat said, 'Let's imagine that we are a query optimizer, and we are trying to figure out some way to process this query.' First we need to find all the employees who earn less than twenty-five thousand dollars. Well, a really good way to do this would be to use a salary index, because using that index you could scan through the data in salary order and, of course, the first people you come to are going to be the lowest paid ones, so that would be very efficient. So using the salary index, we'll go and find the first guy, he's the guy with the lowest salary, and if it's less than twenty-five thousand dollars we'll give him a raise. Let's suppose that he earned ten thousand dollars. So according to this plan, the query processor would find this guy and give him a raise and now he wouldn't earn ten thousand dollars anymore, he'd earn eleven thousand dollars and that would cause him to jump ahead in the index. He would still be in the index, but he wouldn't be in the place he was before anymore, he'd move ahead in salary order. So then, we'd go along and process the next guy and the next guy and pretty soon we'd come to the guy that we gave a raise to already, because now he was ahead in the order, and when we came to him again he'd still earn less than twenty-five thousand dollars, so we'd give him another raise and so on. So this guy might jump ahead in the order of the salary index several different times. In fact, everybody that earned less than twenty-five thousand dollars would get as many raises as they needed until they earned more than twenty-five thousand dollars. And you can see what the problem is here. That obviously was not the intention of the update, but that's the way that it might be naively implemented if a salary index were being used as the access path. As it turns out, Pat and Morton discovered this problem on Halloween.

Frana: Oh, that's why it is called the Halloween problem?

Chamberlin: Yes. It has absolutely nothing to do with Halloween. I remember they came into my office and they said, 'Chamberlin, look at this. We have to make sure that when the optimizer is making a plan for processing an update, it doesn't use an index that is based on the field that is being updated. How are we going to do that?' It happened to be on a Friday, and we said, 'Listen, we are not going to be able to solve this problem this afternoon. Let's just give it a name.

We'll call it the Halloween Problem and we'll work on it next week.' And it turns out it has been called that ever since.

Frana: Everyone was making more than twenty-five thousand dollars in the end, even the ten thousand dollar employees. But you figured that out before people started getting their paychecks I take it? Are there other cautionary tales like that that get passed down?

Chamberlin: There was one interesting problem that Jim Gray and Mike Blasgen discovered when they were working in a lower level of the system called the lock manager—they observed a phenomenon called the Convoy Problem. I'm not going to be able to describe the Convoy Problem to you in detail because I am a language guy and this is much deeper in the plumbing of the system. But the basic idea was that if you were running multiple transactions, each one of which had to operate on several pieces of data, each transaction would acquire a lock and it would process something and then it would release the lock and acquire another lock on something else. And in some cases these transactions would tend to get queued up behind each other—I think Mike Blasgen coined the term "convoy" for this. The transactions were not being processed independently. They'd all sort of run through the data, each one following another one. A database system likes to do many things at once because there are many users and if you keep them all busy then you will get more throughput. And that can be done best if each transaction is independent of every other transaction, so they are all hitting random places in the database and not interfering with each other. What Mike and Jim discovered was that under certain conditions, a convoy could form in which the transactions weren't independent anymore, and that limited the throughput of the system. It was an interesting problem and they wrote a paper about it. [M. Blasgen et al, "The Convoy Phenomenon," *ACM Operating Systems Review*, Vol. 13, No. 2, April 1979, p. 20]

Frana: I can certainly talk to Jim Gray about that in detail.

Chamberlin: Yes.

Frana: Finishing up here, is there a difference between your generation, the generation that started here, and the people that you see come through the door and begin working at IBM Research now? Are there generational differences between people? Do they have the same burning desire to change the world now that computers are ubiquitous? I know you got interested in science because there was this desire for space exploration, this need to reach the moon. What motivates young people now?

Chamberlin: Well, I think if I look at the new hires compared to the folks that got hired with me thirty years ago, the first thing I notice is they are more diverse. There are more women. There are more people of all nationalities than there were thirty years ago. The next thing I notice is that they are better educated. When I joined IBM I was an electrical engineer. I knew a lot about microwave antennas and stuff, which I never got any use out of at all because computers were a brand new thing. It was just barely possible to study computers as a major in those days. In fact, I didn't choose to do that because I wasn't sure it was going to be a recognized discipline. Today, you are much less likely to find people like me who are hired into a research lab to do computer science but whose formal training is in some other field. It is not completely unheard of, but it is

much less common now to find somebody who has no formal background in computer science as an academic discipline.

Frana: And that began in what, the mid-70's? Is that when you really saw the floodgates of computer science grads open up?

Chamberlin: Yes.

Frana: I detected something in what you just said that maybe this is not all for the best. You were not that way, but they are all very formally trained. Is there something to be said for people who don't have the all the trappings of formal training in computer science? Is there a role for these people in IBM Research? Should there be a role for people like that at IBM Research more than there is?

Chamberlin: Well, I think formal training is a good thing. I think that if you are studying algorithms you need to know the state of the art in algorithms and that is what people are learning in universities. Computer science is more of a mature field now. Maybe if you are looking for something today that is like computer science was in the '70's, you may want to look at molecular biology, where revolutionary things are happening and people are racing to map the human genome.

Frana: Is IBM Research picking up molecular biologists?

Chamberlin: Not a lot of them. That is not a major focus for IBM as a company. But I do think that we have some life science initiatives in our database department, trying to provide the infrastructure for these guys to record their data in. Any discipline is going to go through different stages as it matures and probably computer science is a more mature discipline now than it was thirty years ago.

Frana: Would it be fair to say there is more normal science going on now? You know, 'Even IBM Research is working on things that aren't revolutionary the way they used to be.'

Chamberlin: Oh, I think IBM Research is still working on revolutionary things. Relational databases may be a pretty mature discipline, but we've got people in the lab who are working on nanotechnology, and on room temperature superconductors, and on quantum computing—I don't know whether any of these things will pan out, but if they do they could revolutionize the world again.

Frana: It's funny you mention nanotechnology. There is a fellow that runs the Nanotechnology Research Lab at HP (Hewlett Packard) here in Silicon Valley. I don't remember his name [Stan Williams], but he got famous over the summer. Maybe you heard his comment that, 'Everybody under forty-five in my lab was born outside the United States. Everybody over forty-five was born in the United States. We need to somehow attract American students of all stripes and colors back into computer science and not rely so much on international graduates.' He created quite a splash because he gave this congressional testimony- I wish I could remember his name now- and they were all up in arms, until the latest September 11th crisis, about how to get more

students interested in computer science domestically. Do you think there really is a crisis, a real problem? And how do we go about reconciling it if is a problem? I know this is pretty far afield, given what your interests are, but it will have an impact on your life I am sure.

Chamberlin: I don't know if I'd use the word crisis, but I do think that we need to find a way in this country to make science and engineering more attractive to young people. We do seem to be attracting people from all over the world to come here and take advantage of our universities and our opportunities, and that enriches our culture, of course. But I really would like to see lot more young Americans growing up with the idea that science is something that they can do. It's not all that hard. It's challenging and rewarding and I'd like to see more young people choosing that profession.

Frana: Do you think that maybe it boils down to that it *is* too hard, and that selling stuff is easy? It's really not, but it is seen as easy.

Chamberlin: I always thought that selling stuff was hard.

[laughter]

Frana: I have no interest in it myself. I don't want to demean my own generation, but you know, we are accused of thinking, 'Entertain me, I am not worried about what is going on in the box, I just want the entertainment to flow out of it.' Is that something that you think is inhibiting large numbers of domestic students from getting into computer science?

Chamberlin: I don't know the answer to why we don't have more people from our public schools choosing technological careers. I don't think that people are any less intelligent or capable than they used to be. I think the younger generation has all the same talents and abilities as earlier generations. Why aren't they choosing technology as a career? I couldn't tell you.

Frana: Well, here's another very leading question: We entertain this belief in America, I am abstracting, generalizing here of course, that Americans are idea people and we come up with really interesting different ideas. 'Think Different.' And sure, people come from India and they've got perfect SAT scores, but they get a very formalized education, so formalized that they are really teaching to the test. They come to America and they can't come up with the interesting different ideas American kids can. I am not saying that this is my belief at all, but that this is the way it gets represented. So the American kids win because they are able to see beyond the test, beyond the extant body of knowledge that is in computer science. Is that fair?

Chamberlin: No, I don't think it is.

Frana: You have never seen this?

Chamberlin: I don't think that we can even say that the American kids win. If you look around Silicon Valley at a lot of the successful startup companies, many of them are founded by people who have come to this country from somewhere else. There is a very large and successful community of software developers from India here, and from Taiwan. And I think that they are

doing very creative work. In fact, I think that the American school system has a big challenge to keep up with these people. You know, I don't like to see such a large fraction of our technical talent coming to this country from somewhere else. Frankly, these are great jobs. You get to do interesting work, you get to solve problems and have a lot of fun, and get a nice salary and a certain amount of recognition and have an impact on the world - and people should be taking advantage of these opportunities. You don't have to come from somewhere else to do that. People who grew up in California should be moving into these opportunities themselves.

Frana: Well, thanks for going on that excursion. I know that your job is not to change American education. Just to finish off here, I know we are out of time, but what do you do when you are not up here on the hill? I am sure that some future author using this transcript will want to know something about you as a person outside of IBM. Do you harbor deep interests in other things? Are you a woodworker?

Chamberlin: I have built some furniture for our home.

Frana: I am surprised there is not an IBM woodworking shop somewhere in the country as so many of you are woodworkers!

Chamberlin: I like to read, I like to go sailing and camping, and raising two kids is pretty much a full-time job.

Frana: Do you have small children yet?

Chamberlin: No actually, that job is pretty much winding down. My kids are both out of college now. But over the last twenty-five years I have spent a lot of time on that.

Frana: I know this requires a lot of reflection, but is there something you want to convey here that explains you, that makes you, you. Something that you have lived by- basic rules for living and for having a useful career?

Chamberlin: Well, that's a pretty big question.

Frana: It is. It is the biggest question of all.

Chamberlin: I think that I get the greatest amount of fulfillment from building things. It is a very satisfying experience to create something that wasn't there before that will affect the world in a positive way and that a lot of people will use and benefit from. That is what I like about the engineering profession, because that is what engineers do. They create new things. To actually get paid for that is more than you could expect.

Frana: Thank you Don.

End of Interview.

APPENDIX

Relational Databases: Putting the World Online

by Don Chamberlin

Presented at the 15th Anniversary Celebration, Almaden Research Center—June 21, 2001
(Courtesy, IBM)

If you have a bank account or a credit card, you are a relational database user. If you pay a utility bill or make a travel reservation or place a bid at an online auction, you are probably using a relational database. Relational databases are the technology underlying the most visible parts of the computer revolution—the parts that most affect the everyday lives of people all over the world. Databases have become so pervasive that it is astonishing to realize that this technology is only a few decades old. This morning I would like to tell you a little bit about how relational databases were developed, and about the role that was played in this development by some people at Almaden Research Center and its predecessor, the IBM San Jose Research Laboratory.

To tell the story of relational databases, I first have to take you back to a time when dinosaurs roamed the earth, and a typical computer filled up a room and required a special raised floor and air conditioning. Time on a mainframe computer like the IBM 7090 was worth hundreds of dollars per minute in the early sixties, when you could buy a Volkswagen for eighteen hundred dollars. Needless to say, these were not personal computers. My summer job as a college student was to prepare data on tape to feed to one of these machines.

In those days, data was stored mostly on tape and read by the computer using a tape drive. It would take a little over three minutes to read from the beginning of the tape to the end at maximum speed. To read a piece of information in the middle of the tape, you first had to read all the information that came before it, and that took a long time.

Because of the sequential property of tapes, the first database applications were batch-type applications. A permanent file of data, such as a list of accounts, would be kept on a tape, in order by some key such as account number. A batch of updates would be collected, also in order by the same key, often on punched cards. The computer would then read the master tape, applying the updates as it went along, and generate a new master tape. This job might be run once per day or once per week according to the needs of the business. The data on the tape was used only for one application, and was accessed by a program that was specially written just for that application. Data was seen as flowing through the stationary computer and being modified as it flowed along.

In September of 1956 an event took place that changed the database world forever: the announcement of the RAMAC, developed at IBM San Jose. Since the invention of the RAMAC, data has been stored mostly on disk. Inside every laptop and many digital cameras you can find a descendant of the RAMAC. The RAMAC could hold as much data as three tapes, but the most

important thing about it was its random access property. A computer could read data in any order, reaching any record on the disk in less than a second rather than in three minutes as required by tape.

The random access property of disks made possible a revolution in the way information was stored and accessed. Indexes could be created on a disk that allowed records to be retrieved in more than one way—for example, by account number, or by balance due, or by zipcode. Even more important, related records on a disk could point at each other, so it became very fast and easy, for example, to find all the orders placed by a given customer.

Disks made possible a new kind of software called a database management system, or DBMS. The DBMS allowed data to be shared by several related applications, such as accounts receivable and inventory control. By sharing data, applications reduced costs and improved accuracy. Access to the shared data was provided by a standard language, called a query language, implemented by the DBMS.

Since data records on a disk can be connected by pointers, there are many ways in which these records can be organized. Various theories sprang up about the best way to organize data, and various systems were built to implement these theories. In some systems, the data records were organized into hierarchies and in other systems they were organized as networks.

When data began to be stored on disk, it was no longer viewed as flowing past a stationary computer in a linear stream, but as a kind of space that can be navigated by following paths among the records. For this reason, early disk-based systems were sometimes called "navigational." In a navigational system, information is represented by the ways in which records are connected together. To answer a question in a navigational system, you write a program that navigates through the pointer-paths to find the information you want. Several languages were invented for doing this kind of navigation.

Navigational database systems make some kinds of questions easier to ask than others. Questions that are anticipated in advance can be supported by navigational paths that make them easy to express and efficient to execute. On the other hand, questions that were not anticipated may be difficult or even impossible to ask. For this reason, navigational databases place a great burden on the database designer, and they do not easily evolve to meet changing requirements.

In June of 1970, a staff member at the IBM San Jose Research Laboratory named Ted Codd published a research paper titled "A Relational Model of Data for Large Shared Data Banks." Ted was an Oxford-trained mathematician and a veteran of the Royal Air Force. He had a vision about data that was considered radical at the time. Ted believed that people who wanted to retrieve information were not really interested in pointers and navigation. From his mathematical training, Ted knew that all information could be represented by mathematical structures called relations, and that questions could be expressed by a mathematical language called the relational calculus.

One way of thinking about a relation is as a table with rows and columns. Ted Codd believed that all the information in a database should be represented as values in the rows and columns of tables, and that no information should be represented by pointers or connections among records. He believed that queries should be written in a language that is based on high-level, user-oriented concepts like tables. The job of finding a way to process the query should be done automatically by the system rather than by a human navigator. Ted called this new approach to storing and retrieving information the Relational Data Model.

One of the main advantages of the relational data model was something called "data independence." Since queries weren't written in terms of physical things like pointers and indexes, there was no way a query could depend on the existence of these things. Indexes might exist on certain columns of a table, and the system might choose to use a particular index to process a query, but if the index went away, the system would simply choose a different plan that didn't require the index. This meant that the job of writing queries became separated from the job of creating indexes and other access paths. Queries became easier to write because they were written in terms of higher-level concepts. Database administration became easier too because it was possible to change the physical details of the database to meet changing requirements without rewriting all the applications.

Ted Codd's paper on the relational data model became one of the most widely-referenced papers in all of computer science. Ted was named an IBM Fellow in 1976, and in 1981 he received the Turing Award, the highest technical honor in the computing profession. But you'll notice that there was a period of time between publication of Codd's paper in 1970 and his receiving all these awards. Some interesting things were happening during this period of time.

Surprisingly enough, everyone did not immediately abandon their navigational databases and embrace Ted's new relational ideas. Companies had big investments in their existing databases, and many people were skeptical that a high-level relational query could perform as well as a hand-tuned program written by an expert human navigator. This was basically the same debate that had raged over the introduction of high-level programming languages like Fortran, fifteen years before. In order for Codd's ideas to be accepted, they had to be proved by an industrial-strength implementation. IBM stepped up to this job with a project called System R, organized at the San Jose Research Laboratory in 1973. The technology developed by System R was first tested on an experimental basis with a few IBM customers, and it eventually provided the foundation for the DB2 family of products.

One of the parts of the System R project was the development of a new query language. Ted Codd had proposed not just one but two languages, called the relational calculus and the relational algebra. My friend Ray Boyce and I were interested in languages and we felt that the world needed a new relational query language. For one thing, Ted's languages were full of mathematical symbols that couldn't be typed on a keyboard. For another thing, Ted's languages provided a way to query data but didn't provide any way to update the data. Ray and I wanted to fix these problems and also to add some other new features such as grouping, views, and authorization. So we developed a relational query language based on English keywords, called SEQUEL, and published it in a research report in 1974. SEQUEL became the query language of System R. Later it was renamed SQL because of a trademark conflict, and SQL was adopted as

the user interface of DB2 and many other relational products. The language became an International Standard in 1986, and the standard has been updated with new features several times since then, most recently in 1999. The SQL Standard has been important to the growth of the relational database industry because it provides a standard way for different systems to talk to each other.

Another thing that was happening at about the same time was a study of a database design principle called normalization. It turns out that if you decide to represent information in the form of tables, not all ways of organizing the tables are equally good. To illustrate this point, imagine a table that contains information about employees and departments. Some information, such as salaries, belongs to individual employees. Other information, such as floor, belongs to departments. A better database design would factor out the department information into a separate table. In this database design, the location of each department is stored only once instead of being repeated for every employee. This is an example of a process called normalization, which has been extensively studied by several people at this laboratory, including Ted Codd, Ray Boyce, and Ron Fagin.

Ray Boyce worked with me on the SEQUEL language and with Ted Codd on a form of normalization called Boyce-Codd Normal Form, which is still taught at universities today. Ray died suddenly of a brain aneurysm in 1974, shortly after the publication of the first SEQUEL paper. In his short career at IBM, he made a mark on the database industry that is still very visible nearly thirty years later.

As I mentioned earlier, the key problem that had to be solved for relational database systems to be successful was to find a way to automatically translate a high-level query into a detailed plan for executing the query. A one-line query in a language like SQL might translate into a plan that includes several nested loops, uses one or more indexes, and performs complex operations like sorting. The challenge was to automatically generate a plan that executes as efficiently as a plan written by a skilled human. This is a very difficult problem because of all the variables and choices that need to be made. The part of a relational system that deals with this problem is called the optimizer, and the person who designed the optimizer for System R was Pat Selinger. Pat created a mathematical model to predict the cost of various ways to process a query and to generate the least expensive plan. This approach is called cost-based optimization. Pat built the world's first cost-based query optimizer, and the techniques that she invented are still taught in university computer science courses today.

Pat Selinger's optimizer gave people some confidence that relational systems could perform as well as navigational systems. For her work on cost-based optimizers, Pat was named an IBM Fellow in 1994 and elected to the National Academy of Engineering in 1999.

A query optimizer can make up a good plan for processing a query, but it might take some time to do so—maybe half a second or so. This is OK if you are only going to ask the question once. But what if you are the MasterCard company, and you want to look up whether a credit card number is valid and has enough credit balance for a new purchase? What if you want to answer this question hundreds of times per second as credit checks are phoned in from all over the country? It's the same question every time—only the account number changes. Obviously you

can't afford to run every credit check separately through the query optimizer to choose a new plan from scratch every time. What you need is a way to optimize the query once and save the plan so it can be executed repeatedly. The person who figured out how to do this was another member of the System R project, Raymond Lorie, who published a paper about it in 1981.

Raymond's query compiler proved that relational systems could be used not just for ad-hoc queries that were executed only once, but for high-volume transactions as well. This was another piece of the puzzle that positioned relational databases to take over the world of commercial data processing. For his work on compiling relational queries, Raymond was named an ACM Fellow in 2000.

Automatic teller machines are found all over the world. If you insert your plastic card into one of these machines in Tokyo, out will come some yen. If you insert the same card into a machine in Amsterdam, out will come some guilders. A month later, you will receive your charges itemized in dollars in your regular statement from your home bank. We think of these machines as part of the natural landscape. We expect them to work, and they do. In fact, these machines are only the most visible part of a network in which billions of dollars are exchanged every day among financial institutions all over the world. Electronic transactions have become the basic mechanism by which value is exchanged and resources are allocated in our civilization. Underlying this network are relational databases and some surprisingly subtle ideas that were first investigated in a rigorous way here at IBM.

Providing efficient access to data for one user is hard enough. But when thousands of users are sharing the same data and trying to read and write it at the same time, a whole new category of problems arises. Each user needs a guarantee that his or her business will not interfere with the business of other users who are accessing the database at the same time. The study of this kind of guarantee and the technology that underlies it is called the theory of transactions.

When you take part in an electronic transaction, you take certain things for granted. For example, if you move funds from one account to another, you expect all the parts of the transaction to take effect at once. You would not be happy if money vanishes from one account and fails to appear in the other. The property that you are taking for granted is called Atomicity.

Another thing you take for granted is that, at the end of a transaction, all your records will be in a valid state. Exactly what this means depends on the transaction. It might mean that your credit limit is not exceeded and your checking account is not overdrawn. The property of a transaction that makes sure it does not violate any rules is called Consistency.

A third thing that you take for granted is that, even though thousands of people in many different locations are all making updates to the same shared data, the computer will not get these updates mixed up. It will not sell the same airplane seat to two different people. If you are in Tokyo and your wife is in Amsterdam and you both try to withdraw your last one hundred dollars at exactly the same time, one of you will get the hundred dollars and the other will get a message saying that your account is empty. This transaction property is called Isolation.

A fourth thing that you take for granted is the property of Durability, which means that, after your transaction is confirmed, the computer will not forget about it. Once you get the slip of paper confirming your deposit, that deposit will not vanish from the bank's records, even if there is a rolling blackout in the very next second. This property is especially important in California.

The four properties of electronic transactions, called the ACID properties, were first investigated in a rigorous way by Jim Gray at the IBM San Jose Research Laboratory. Jim helped to develop the technology that underlies these properties, including protocols like "two-phase commit" and "write-ahead logging." Jim literally wrote the book about electronic transactions. For his work on the theory of transactions, Jim was named an ACM Fellow and was elected to the National Academy of Engineering in 1994. In 1999, Jim received the Turing Award, the second Turing Award to be given for database work done at IBM San Jose.

By the early '90's, relational databases had grown so large that they exceeded the capacity of a single machine, both in terms of data volume and transactions per second. The only way to keep up with the growing demand was to find some way to spread the data over more than one machine. There are several ways to do this. Several processors can be put in a box and given access to a shared set of disks. Or, several computers on a local area network can divide up the data and cooperate to process queries. Or, carrying the idea of distributed data to an extreme, a computer in San Francisco might ask a computer in London for some help in processing a query. Each of these kinds of parallelism has its advantages and its challenges that need to be solved.

Bruce Lindsay of Almaden probably knows more than anybody else about how to distribute data and process it in parallel. Bruce worked with Pat Selinger to develop a protocol called DRDA that enables DB2 servers all over the world to talk to each other. He developed algorithms that allow databases to scale efficiently to very large sizes by adding more processors. For helping to remove the limits to the growth of relational databases, Bruce was named an ACM Fellow in 1994 and an IBM Fellow in 1996.

If you're running an electronic business like eBay or Amazon, you really don't want any of your data to be lost, no matter what happens. You need a backup copy of your data, but you can't afford to shut down your system while you make a copy. You need to make sure that if anything fails, all your completed transactions will survive and your data can be restored to a consistent state. In short, you need to anticipate all kinds of hardware, software, and power failures and make sure that you can recover from any of them without losing data, no matter when the failure happens. You must do this while your system is processing thousands of transactions per second, and you must do it without degrading the performance of the system. This is called the database recovery problem, and it is one of the most difficult problems in computer science.

The state-of-the-art solution to the recovery problem was developed here at Almaden by a researcher named Mohan, who published it in the Very Large Data Base conference of 1989. Mohan's algorithm, called ARIES, is now used in DB2 and several other IBM products, and taught in universities around the world. For his work on database recovery, Mohan received the SIGMOD Innovation Award in 1996 and was named an IBM Fellow in 1997. Ten years after the publication of Mohan's paper on ARIES, it received a special award for its enduring significance and impact on computer science.

With the success and widespread use of relational systems came a new set of problems. Customers began wishing for a long list of new features. A lot of these new features were added to the SQL Standard, which grew to an impressive size. IBM had to support these new features not just on one operating system but on OS/390, Unix, Windows, and several other platforms. What was needed was a way to make DB2 more extensible so that new features could be added more easily. Here at Almaden, a project called Starburst was figuring out a way to do this. At the heart of Starburst was a versatile data structure called Query Graph Model, proposed by Hamid Pirahesh. QGM and other Starburst technologies were transplanted into DB2 in the mid-'90's, in a massive technology transfer involving Almaden and several other IBM laboratories.

Hamid's QGM data structure made it possible for DB2 to support the advanced features of the new SQL Standard. It also made possible some new query processing techniques that improved the performance of some queries by a factor of 10 or more. The QGM transplant gave DB2 both industry-leading functionality and industry-leading performance. Hamid's impact on DB2 was recognized when he was named an IBM Fellow in 1999.

My first day of work at IBM was in the same month that Ted Codd published his famous research paper introducing the relational data model. Relational database systems are now a seven billion dollar-per-year industry, and DB2 is one of IBM's most successful software products. Relational systems store most of the world's online business data, and handle billions of transactions per day. Along the way, a number of difficult problems were encountered that could have been show-stoppers, but these problems were overcome by creative and resourceful people, including the people I've talked about today. The people of Almaden have had a lot to do with creating the relational database industry and establishing IBM's leading role in it. I've told you about only a few of these people, but there are many more that I wish I had time to mention. It has been a great privilege for me to spend my career working in this place and with these people. Thanks for joining us in our celebration.