Oral History of C. J. Date

Interviewed by:
Thomas Haigh

Recorded: June 13, 2007
Mountain View, California

CHM Reference number: X4090.2007

Table of Contents

C. J. Date

Conducted by Software Industry SIG—Oral History Project

Abstract: C.J. Date (Chris), the database management system specialist and writer, talks about his early background and education, his joining IBM and his initial work in programmer training at the Hursley Lab. This was followed by work on database management under PL/I which led to his involvement in the relational data base concept. He then talks about producing his first database book and how it evolved over time. This book was followed by many additional database textbooks and papers. He had developed a relationship with Ted Codd early in his career and, after both left IBM, they started the Codd and Date companies. Finally, he describes his work with Hugh Darwen on *The Third Manifesto,* which presents his view of the foundation for future database systems.

Thomas Haigh: We are here at the Computer History Museum in beautiful Mountain View, California. The interview is taking place during the afternoon of Wednesday, June 13, 2007. The oral history interviewee is C.J. Date, the well known database management system expert, writer, and consultant, and I'm Thomas Haigh, a historian. This oral history is being conducted under the sponsorship of the Software Industry Special Interest Group of the Computer History Museum. We have already reviewed a general list of the areas to cover. One thing I would encourage you to do: if you mention people's names, please spell them out to make it easier for the transcriber.

Chris Date: Okay.

Haigh: Also, if you're mentioning a particular book or paper, please give enough information so that we can reconstruct the citation for it.

Date: Okay.

Haigh: We have just two hours, so we'll have to prioritize some topics over others.

Date: Can I just say that I never use the word "expert." Specialist I will accept, but <laughing> no one's an expert, you see.

Background and Education

Haigh: I wonder if you could begin by saying a little bit about your early life and family background.

Date: Sure. I was born on January the 18th, 1941 in Watford in England and lived on the edge of London as a kid in the early days of World War II, which means my earliest memory is actually hearing bombers coming over and that sort of thing. My father was away in the war. My mother was part of the local team who went to put out the fires when incendiary bombs were dropped… interesting early memories. Anyway when I was seven or eight, or something like that, we moved to a country town, Aylesbury in Bucks in England. And I always think of myself as a country person, even though I was brought up in the town. I like the country, and I live in the country now. I went to primary school in Aylesbury. Then I went to High Wycombe Grammar School, which is a few miles away. I was a boy, so I was interested in science. And in school, I specialized in scientific subjects, which meant really mathematics, physics, and chemistry. It so happened we had a terrific math teacher and rather poor physics and chemistry teachers, so I became a mathematician. So when I went to university, mathematics was what I studied. Is that as much as you need?

Haigh: Sure. As a boy, did you enjoy hands-on technical projects?

Date: No, probably not. I was always more of a cerebral type. I like to read, voracious reader, and I have more books in my house than you can count. And I like to think about things, and I like art, paintings – things like that. But playing with gadgets, that sort of stuff, I never did that.

Haigh: And did you always have the expectation that you would go to university?

Date: No. I was the first one in my family to do so. But when I was at the grammar school I discovered I was doing very well at school, I was at the top of the class, and it was sort of effortless. And I wasn't working hard, so it was fine. So when it came to applying to university, I thought there was a good chance I would get in, but it was a novel thing. And I did get in and discovered that no longer was I a big fish in a small pool – I was a very small fish in a very big pool, and suddenly it was very hard work.

Haigh: At that time, did you have any kind of sense when you arrived at university of what you wanted to do with your life?

Date: Not at all. I knew I was going to study mathematics, and at that time, computing was very new. This was 1959. There was no computer science education in college at that time. So when I was getting close to acquiring my degree and the question was what do I do

now, the university had an appointments board, which would help place people in jobs. So I went along to see them, and they said, "Well, okay, you're a mathematician. Here is a list of jobs mathematicians can do." And they mostly sounded desperately boring. Time and motion study was one of them. But then there was computing, a new one. And I said, "Well, I know nothing about that. Let's have a look at that one." So they arranged some interviews with computer companies.

Haigh: Before we get into that, I'd like to backtrack slightly. You didn't go to just any university. You went to Cambridge. Did your teachers encourage you to apply there?

Date: There was a kind of a connection between the school and some of the colleges at Cambridge, partly through the teachers themselves having been there and so on. We had a track record of sending one or two people a year to Cambridge, so I think it's fair to say that most people applied to Cambridge rather than Oxford from our school. And our year was no different; I think we got four people accepted that year, something like that, into Cambridge.

Haigh: And which college did you attend?

Date: College in Cambridge is really a residential sort of thing, so it was Jesus College.

Haigh: You say that when you got there you found that you weren't the big fish in the small pool anymore. Did you find the mathematics education there challenging?

Date: Very much so, and I made a horrible mistake. I got an open exhibition, which is like a junior scholarship, to get in. And they said, "Okay, you must be very smart. We'll start you in a second year course in your first year." So for a semester, I was doing a second year course and not understanding anything. So at the end of that time, I realized I should drop back to the first year, but I missed the first term of the first year, and I never really got over that. I finally got my degree, but it was a skin of the teeth sort of job. It was not a good degree.

Haigh: Were there any areas of mathematics that you found particularly interesting?

Date: Yes. The more abstract the better. I loved the abstract algebra stuff, which of course became very relevant many years later.

Haigh: Now at that point, did you have any awareness while you were there that there was such a thing as a computer and indeed that Cambridge was one of the centers for computer development?

Date: I think I would probably have to say no, other than the stuff you saw in the popular papers. I knew there were things called computers. But I knew nothing about them first-hand at all. No one suggested it to me, none of my professors or anyone like that suggested that maybe this was something that was pertinent. [Which is a little odd, now I come to think of it. My Director of Studies at Cambridge was Derek Taunt, and I learned only recently that he had been one of the mathematicians at Bletchley Park during the war, working with Alan Turing. So he must have had first hand knowledge of the Colossus, which is often regarded as a precursor to the modern computer. And he must have had some idea of what Turing was thinking about at that time, and later he must have had contact – presumably continued contact – with the people building the EDSAC at Cambridge. But as far as I know he never mentioned any of these things to any of his students. Of course he would still have been under all kinds of security constraints when I was at college, even though the war was well in the past by then.]

Haigh: So at that point, at least as far as the students were concerned, there was no real connection between the mathematics department and the computer department?

Date: That was certainly true in those years. And I should say something else too, which is that although I was there to study mathematics, I guess I didn't learn as much mathematics as I should have done. I learned a lot of other things. I met people in other departments. I became very interested in music then, for example, since we never had music at home. So I learned a lot of things at Cambridge, but I didn't learn very much mathematics <laughing>.

Haigh: Now, you're well known as a writer. During your childhood and undergraduate days, had you enjoyed writing things?

Date: Yes. I always liked writing, even in school, doing school compositions, I always enjoyed that.

Haigh: And was that unusual among the mathematics students?

Date: I really couldn't say.

Haigh: Before we move on, you still refer to yourself as a mathematician sometimes.

Date: I say I used to be a mathematician.

Haigh: All right, you used to be a mathematician. So what elements of that training do you think have really shaped the way that you addressed the rest of your career?

Date: I think there are two things. First of all, the theory of sets and related stuff was directly relevant to the relational model when it came along some years later. That was a specific application. But then of course, there's just the general training and the way of thinking about things, trying to analyze problems and think about things very clearly and figure out what the crucial issues are. It was a good mental discipline.

Haigh: And did you ever consider doing a master's degree or a PhD?

Date: Yes. <laughing> I have thought about it, but I would feel a little fraudulent in a way, because my first degree is not very good. What I would really like to do is go back and do my first degree properly. That's not going to happen, but a curious thing has happened. Since I wrote that first textbook, it's now used in many colleges and universities – often for courses for which I don't have the prerequisites. So I couldn't attend the courses themselves in principle.

Haigh: You've mentioned that when you finished your degree you still hadn't decided what to do, and the career guidance people had mentioned the possibility of computing.

Date: Yes.

First Job in Computing: Working for Leo

Haigh: Did you interview with more than one company?

Date: There were at least three. The two main ones anyway were Leo Computers, which at that time was a British company, an offshoot of J. Lyons, the Tea Shop Company. And I interviewed with ICT. Much later of course, ICT became ICL, and Leo was also a part of ICL eventually. Both of these offered me a job, but there was a difference. The ICT offer was conditional on my gaining my degree, and the Leo one was unconditional, so I took the Leo one. And also, I liked the feeling at Leo when I went for the interview much better than ICT. An example is, when they sent the letter for the interview, they gave the address and the time and so on. And the letter said, "We are quite easy to find. We are opposite Sketchley's the cleaners." Somehow that was an ad hoc British sort of way of doing things which appealed to me, so when they offered the job I took it. And that was a great place. They had first built the Leo I, which was used in fact to do the U.K. census in 1951. It was one of the first commercial [electronic computer] applications in the world. Then, they built a range [of machines] called Leo II. There were a few machines there. I came on board at the time they were developing Leo III, which was another range of huge machines. But the team, the software people I was dealing with, they were just wonderful. They had a high level language that they had developed themselves and they were leaders in inventing compiling techniques. They had an operating system they called the Master Program. They were inventing new techniques there. And in the machine itself, they had invented all kinds of things. For example, they invented channels; they

invented multiplexing; they invented index registers. And everyone was having such a great time doing this stuff. No one wrote it down. We didn't go around writing technical papers and submitting them to conferences. We were just having fun doing it. And of course, it was kind of frustrating when you look back later on. Other people took credit for these things, because they did write them down.

Haigh: Was it 1962 when you graduated and arrived at Leo Computers?

Date: Correct, yes.

Haigh: Were most of the software people they hired mathematicians?

Date: No. As a matter of fact, very few of them were. They looked especially for linguists. They felt that people who were good in languages, people who were good at expressing themselves, were therefore people who were good at thinking clearly. And the old cliché about do you like doing crossword puzzles, that was one of the questions that was asked. And there was an aptitude test you had to take before you could be offered a job. I used to administer that test later on. And the way that worked was, it was a full day thing. In the morning, you would have this class of people, and you would teach them elementary programming in a hypothetical machine level language. And in the afternoon, you gave them a test of 10 questions, and they had to write little programs, in effect. So you were directly testing native programming ability, which was an interesting concept, because you couldn't assume people had done programming before. Anyone who would do well on the test was probably going to be a good programmer. We might have missed some good people, but we certainly got very good people.

Haigh: And after you were hired, did you go through a programmer training course?

Date: Yes, a five week course.

Haigh: And can you remember if you immediately enjoyed programming? What was your personal reaction?

Date: Programming was great fun, yes. Although, in those days, I preferred programming at the machine level, the assembly level, over high level stuff, which is ironic considering what happened later.

Haigh: You described your work there as a programmer and programming instructor.

Date: I was hired as a programmer, and in fact a mathematical programmer. There was a mathematical programming department, and I worked on a problem called the Transportation Problem, which I would have to take a while to reconstruct exactly what the problem is, but it doesn't matter now.

Haigh: Was that some kind of operations research style optimization program?

Date: Yes, exactly right, yes. And I was doing that for a few months. But then, they were very short of staff in the training department. So a few people, and I was one, were sort of pitch forked out of what we were doing into the training department. And this is really strange, because if you had said to me that I was going to make a large part of my living by standing up and talking in front of a group, I would have said, "You're crazy, no way. I couldn't possibly do that." But I was thrown into the deep end. I was given a set of lectures to do, and of course, the very first one I had to do, I'd been given an hour and a quarter for the session, and I finished in half an hour. And I thought, "Oh my God, I got this all screwed up." It turned out it was a half hour subject. They just had the timing wrong. The sensible thing was that I didn't just do one lecture. I had a course of 10 to do, so by the end of the 10, I was feeling quite comfortable with it. And I discovered that I really liked it, and I discovered I was quite good at it. I'm not supposed to say things like that, but it's true. And so I stayed on in the training department for quite a while; I was still learning stuff but also teaching.

Haigh: And were you training employees at the user companies or just the Leo employees?

Date: Both. We didn't make a distinction. The classes were mixed.

Haigh: And you were there for five years?

Date: Yes.

Haigh: And continued to work as an instructor?

Date: Most of that time, but that also included my writing reference manuals and so on. That was regarded as a training department function in that company at that time. And I had fun with that too, because I liked writing and trying to describe things very clearly and carefully and precisely and accurately. I got a kick out of doing that.

Joining IBM

Haigh: Then, in 1967, you went to work for IBM.

Date: Yes. There was an ad in the paper. IBM Hursley wanted instructors, and the great thing about this was that they were located in the countryside, and I was working in the town. Although, to tell you the truth – I went for the interview; I took the day off and went down – I had no intention of accepting a job, even if it was offered, because IBM had a terrible reputation, as you may know, in the U.K. at that time. Basically, it was regarded as a very sales oriented company, an American company. And I was working for a British company and a more amateur kind of thing. And to my great surprise, when I got there not only did I think the physical surroundings were great, it was near the town of Winchester, which is beautiful, and it was in the countryside, the people were great too; there were very, very good people there.

Haigh: So how would you compare the cultures of the two companies?

Date: Well, I don't think, at that time, I could talk about an IBM culture. There was a culture in the Hursley Lab, which I liked very much. It was very informal, very smart people and nice people. I learned later that different parts of IBM had different calibers of people.

Haigh: And the Hursley Lab had a distinctly British kind of culture to it.

Date: I don't think it was British. I remember my shock when I learned that everyone measured prices in dollars. So no, it was clearly part of an American company. That was not a problem. It's just that they were very good, interesting people, with lots of other interests besides computers. So you could have intellectual conversations that were fun.

Haigh: And you were initially doing the same kind of training work there as at Leo?

Date: Well, two of us were hired in, both of us from Leo, to establish basic programmer training in the IBM Hursley Lab. And almost the first thing that happened to us was we were sent to IBM Poughkeepsie to attend basic programmer training, not to learn the material so much, because we basically already knew that, because by that time Leo was building some compatible machines. It was not to learn the material but to see how it was taught, and we saw how not to do it, and we came back and did it a whole different way.

Haigh: Was that your first visit to America?

Date: Yes, it was. That was 1967, I guess.

Haigh: So what was it that you didn't like about the Poughkeepsie teaching style?

Date: Well, we happened to have a very bad instructor. He was so bad that the education department manager called us out one day into his office and he said, "I know why

you're here, and I want to say don't judge the whole education department by what you're seeing this week." I don't want to say any more about that, but we had our own ideas about how we should do it. And we came back to Hursley, and we set up our own structure for the classes, and I flatter myself that they were good, and we ran those for some years.

Haigh: You mentioned in your note that you had been teaching computer systems fundamentals, 360 Assembler, and PL/I.

Date: Correct. Hursley was the home of PL/I at the time, and so those were the three things that we taught at first.

Haigh: Did you like PL/I?

Date: Yes and no. Not an elegant language, but it was effective. You could do anything you wanted to do. A friend of mine once said to me about PL/I, "The one thing you cannot say it is, it is not the language of least astonishment." <laughing> You could do what looked like very simple things and get extraordinary results. There were some traps in the language. It's possible, for example, in PL/I to construct three variables, A, B, and C with the property that A greater than B is true; B greater than C is true; and C greater than A is also true. <laughing> So you had to take care. But yes, it was effective. It was an effective language, but not academically elegant.

Haigh: And were you still training a mixture of IBM employees and customers?

Date: No. The mixture was in Leo. In IBM, it was different. They kept the employees and the customers separate. We were only doing in-house training.

Haigh: And you also mentioned something called the IBM European Laboratories Integrated Professional Training Program (ELIPT).

Date: Right. IBM had several development labs around Europe. The Hursley Lab was in England; there was one in Germany, one in France, one in the Netherlands and so on. And the idea was to get all these labs together into a scheme where each lab would contribute one or two instructors who would then travel around to all the labs teaching basically the specialty from the home lab (at least that was the original idea; it grew from that). So one particular person was charged with that responsibility. He was our manager, Mike McMorran and myself, the two guys who were hired from Leo. And so together with our counterparts in the other labs, we put together the structure, and Mike and I were two of the first instructors in the deal.

Database Management and PL/I

Haigh: And then, it was at some point during your time at Hursley that you first became aware of the idea of database management.

Date: That's right. That would have been in 1970, because what happened, at least at IBM in Hursley, I can't speak for other locations, but they had this very sensible idea that you can't just be an instructor. You had to do more than that, so we would rotate out of the department. I should say, by the way, that by this time we had several instructors, not just the two of us. So it became my turn to rotate out; I joined the department at Hursley called Advanced Technology, which was like a mini research group inside the development lab. And I was told we are responsible for PL/I; there's this new thing coming along called database management; you go and figure out what we should do in PL/I about database management. And so I read everything I could find. IBM Hursley had a very good library and I read all the technical papers I could find. I played with IMS, which then was the IBM database product. I read the CODASYL database specifications, which were appearing about that time, and wrote toy programs in that language. And Ted Codd's paper came out in June of that year, the Communications of the ACM paper, the famous paper, "A Relational Model of Data for Large Shared Data Banks." And I read that; I said, "Ah, this is sets. This is intersections and unions and things. I know about this stuff. I did this at college." And it was such a simple idea; although it didn't look simple the way Ted had presented it. It was a simple idea. And I said, "This is great."

So I came back and said to the PL/I folks, "This is what we've got to do. We've got to implement relations in PL/I." And I, with a colleague (Paul Hopewell), devised a language as an extension to PL/I to do that. Curiously enough, it looked not unlike SQL, because to digress for a second, the key notion in SQL was they didn't want to do Ted's relational algebra. They didn't want to do Ted's relational calculus. They wanted to do something different. And I didn't say this in the meeting yesterday, because it wasn't appropriate, but I'll say it now. I think that goal was based on a fundamental confusion between syntax and semantics. It's true that if you look at Codd's papers, the algebra and the calculus look very obscure and abstract, but that's just notation. It's very easy to wrap nice syntactic sugar around those ideas and come up with something very palatable. A living example of that is Query By Example [by Moshé Zloof], which is just relational calculus but with a nice syntactic sugar coating on it. Anyway, I digressed a bit there. We came up with the same idea that SQL did, that you have in effect table expressions, they called them subqueries, and then you have a fundamental operation of set inclusion, testing whether two sets are equal or one is contained in another and so on. So Paul Hopewell and I came up with this language, and I still have some of the documents at home describing this stuff. And we built a little interpreter to show that it worked and we did all of the usual examples. And the PL/I people were comparatively happy with this idea. But I'm getting ahead of myself, so I think that's enough on that topic for the moment.

Haigh: And was your idea here that this would be an extension to the PL/I language itself, or did you see it as a separate language embedded within other languages?

Date: We saw it as an extension to PL/I. Perhaps we were thinking too narrowly. We weren't interested in FORTRAN and COBOL and so on. So the idea of having a common language that could somehow be attached to all the high level languages, I think it's true to say we didn't even think about it at the time.

Haigh: And I think that's the same philosophy that the CODASYL people adopted, that in principle there was going to be a COBOL data manipulation language and a different one for FORTRAN and a different one for PL/I.

Date: Correct, yes.

Designing a Database Language

Haigh: So you said you and Paul designed the language?

Date: I should have said also, not only did we make this decision, we got in touch with Ted Codd in IBM Research in California, because we had some questions about his ideas. And so we started up a correspondence, and I still have those letters, too.

Haigh: And was Codd pleased to hear from you?

Date: Absolutely. I didn't realize it at the time, but he was very much a loner and fighting a lone battle, and any evidence he could find that supported his ideas, he was obviously very interested in.

Haigh: And had he been making an active kind of effort to find IBM product groups to support him, or had he just been publishing his papers and hoping someone would read them?

Date: I don't know that I have a definitive answer to that. My perspective is that his papers appeared as it were out of the blue, and we suddenly learned about this guy called Ted Codd. A little later, he certainly was evangelizing for the ideas, but at that point, I couldn't say.

Haigh: At this point, did you have any strong awareness of the kinds of database applications that users were attempting to construct?

Date: I think the short answer is no. We just knew that there was a requirement for databases. We knew that there were different approaches being proposed, and one of them

looked clearly superior. So it became a bit of an abstract intellectual exercise, but we knew there was a requirement. We didn't know exactly what it was, but we knew it was needed.

Haigh: So you wrote this language up and submitted it to the relevant group. What happened to the project?

Date: Of course we wrote it up, and then there were several alterations. As I say, Hursley had the PL/I mission within IBM, so we talked to the PL/I people in Hursley and had some exchanges. And they told us why things wouldn't work, and we made changes and so on. And we also sent the documents to other people like Ted and to other places around the company and got, at that time, mostly positive feedback, I think it's fair to say; however, I probably should say too that most of the rest of IBM really didn't care about PL/I very much!

Haigh: So did you have in your head an idea of where, if it had been adopted, the actual database engine to sit underneath the language was going to come from?

Date: I can't recall that we really discussed that very much. I mean, we knew it had to be something new. There was no question of making this run on top of IMS or anything. We realized that. We were not the home of data management. We were the home of the PL/I language. It was part of the compartmentalizing that goes on in IBM (or used to then, anyway).

Haigh: And I think you also mentioned in your bio that at some time around this period you were active in the British Computer Society working group that was looking at databases.

Date: Yes, that's right. I forget exactly how it started, but it was a special interest group basically on database stuff. And Paul and I, since we were actively working on this relational thing, became sort of the de facto relational experts as it were in this group. We were educating the rest of the group. And then somebody – it might even have been me; I don't remember – but somebody came up with the idea that we should run a British Computer Society symposium. This was an interesting topic; probably lots of people would like to hear about it. And we set up a day (April 5th, 1973), and I believe it was the first public symposium anywhere on relational stuff. There had been ones inside IBM and possibly elsewhere, but this was the first public one. And we invited Ted over as our star speaker. And I did the opening tutorial, and we had some other speakers. I still have the attendance list for this event, and it reads like a Who's Who of British computing at the time, many names in there who subsequently went on to make major contributions. Maurice Wilkes was there, by the way, the grand old man of British computing.

And a few funny things happened. The night before the event, IBM management took Ted and me out to dinner. And the purpose was to shape our heads, so that whatever we said about this relational stuff, we must not do anything that could affect sales of IMS. And okay, we

understood that. And the next day, we proceeded, and at the very end of the day, there was a question and answer session. Ted gave the final session, and various questions were asked, and Ted handled them. And then somebody got up and said, "Dr. Codd, tell me, what does this relational idea of yours do to IMS?" And the whole room went completely quiet. <laughing> You could have heard a pin drop. And Ted stopped and he looked at the questioner. He took off his glasses, and he looked at him again. And then he said, "I did not fly 6,000 miles to talk about IMS." And the audience applauded, and you could hear the IBM guys going Whew! <laughing>

Haigh: But at this point, 1973, it's not just that IBM doesn't have a relational product. Nobody had a relational product. In fact, as I understand it, nobody even had a prototype.

Date: Actually, there was a prototype. The IBM U.K. Scientific Centre in Peterlee built a system which they called IS/1, and that was operational by 1971. IS/1 stood for Information System 1, and the powers that be in IBM made them change the name, because that sounded too much like a product. And they changed it to something that was deliberately unmemorable. They called it PRTV, Peterlee Relational Test Vehicle. But yes, that was running, and it was written in PL/I by the way. So there was a prototype, but there were no products. That's certainly true.

Haigh: I think it's interesting that the meeting would draw such a crowd for an approach that was still so experimental. It wasn't as if everybody had been playing with this in their lab and wanted to get together and see what the latest developments were.

Date: The purpose of the day, of course, was education, telling people what this was. And what it was was an abstract model for the way database systems should become in the future, trying to decide ahead of time what you want to do, instead of just writing code and seeing what happens.

Haigh: But given the ultimate slowness with which relational technology took over the world, it's interesting that right at the beginning, at least this room full of people was extremely open to it and to just come to the meeting, must have seen the potential very quickly.

Date: Yes. Well, part of the slowness was internal politics in IBM, of course. And we've heard about some of that these last two days, so it did take a long time. And of course, there were serious technical problems to be solved, but I do believe it need not have taken as long as it did.

Haigh: You mentioned Maurice Wilkes. Were the other people at the meeting all from universities and research laboratories?

Date: No, people were from consulting companies, some were customers, and one of our presenters was from Esso Petroleum, as I remember, Peter Prowse. Because one of the nice things about the relational stuff was that you could use the ideas to help you design your databases and even design your transactions and so on, even though eventually you would have to map your abstract relational designs into whatever your target system supported, be it IMS or Total or whatever. It was an intellectual tool, even if it existed only on paper, and people could see that.

Haigh: So in practice, that would mean minimizing redundancy.

Date: Things like that, yes.

Haigh: Okay. And was that the first time that you met Codd?

First Contacts with Ted Codd

Date: No. The first time I met him was 1971, I think it would have been. After the correspondence I mentioned before, Ted invited me to come over to the U.S. and present the work that I was doing with Paul Hopewell at various locations in the U.S. I went to four locations. The crucial one though was San Jose, because that was the home of data management. And I felt pretty scared about it too, coming from some remote location in IBM to the center of the world as far as data management was concerned and basically telling them how I thought they should be doing their job is what it came down to. But it went very well, and of course, it was part of Ted's effort to persuade IBM in general that there were other people in IBM interested in his ideas. I didn't know that was his agenda at the time, but that's what it was. And so that's when we first met.

Haigh: And what impression did he make on you in person?

Date: Oh he was a very dynamic guy, and very inspiring. What I particularly liked was that he was very careful in what he claimed. He never over claimed. He thought the relational model was wonderful, of course, but he said, "I'm not saying it's the be all and end all. You never know what's around the next corner. And I'm not saying it can solve all problems, but I do say that the class of problems it can solve is very large." He would not say that this was the silver bullet that's going to solve everything, and I was impressed by that, because so much of what you hear in computing is hype.

Haigh: And how did your relationship develop during the time that you were based in England?

The Introduction To Database Systems Book

Date: We continued to exchange letters and so on, and then I conceived the idea of writing this book.

Haigh: When did you conceive the idea?

Date: Well, it probably was 1971, because I wrote it in 1972. And again, I talked to Ted about it. Well, first of all, I floated the idea to Ted, and he wrote back and said this was a great idea. Again, I have the letter at home. He said something along the lines of a book such as the one you describe is very much needed – and by the way, you can use any of my research papers as material in the book, and all that sort of thing. So our correspondence continued, and I will probably come back to the book in a moment.

Haigh: And at that point, did you conceive it as a book about the relational approach?

Date: No.

Haigh: It was always going to cover the three different database models?

Date: It was a textbook on database management as we knew it at the time.

Haigh: Right. Now, had you been teaching a course on database? Did this, at that point, make it into your instructional work?

Date: Well, that's right. After I had done this stretch in the Advanced Technology Department, I then rotated back into the Education Department and developed a course on database, which I taught as part of this ELIPT Program around the European labs. I did that for probably a year or two; I don't remember exactly. And in developing that material, I wrote an IBM report on the CODASYL stuff. I had to learn it myself, and the specifications are very hard to understand, so I thought that this would be a useful service, and I did that. And after writing that, I said, "Okay, I could write a book about this and IMS and Ted's ideas." And so that was the idea, to describe everything. And as a matter of fact, if I can blow my own trumpet for a second, the division of the database world into relations, hierarchies, and networks was largely my idea, trying to impose some structure on something that didn't have too much structure ahead of time, purely for tutorial reasons. But most of the textbooks that came after did exactly the same thing; they followed that lead. And I realized with hindsight that it was a horrible mistake. <laughing> Can I talk about that?

Haigh: Yes.

Date: In the early days of relational, we were constantly getting into debates. We had to try and demonstrate that in a sense, relational was real, or could be real. And so we had to show that it could solve problems that the hierarchical and network approaches could already solve. So we necessarily got into comparisons between relational and the other approaches. And in getting into those comparisons, unwittingly we promulgated the idea that these were just three different ways of doing the same thing, and in a sense they were equal to one another. But I realize now that was dead wrong. Relational is very fundamental. The other things are just ad hoc solutions, implementation techniques, ways of doing things on the machine, but exposing those ways to the user. The relational model was very different, a very abstract thing, as I characterized it before. A good way to look at it is like this. What is a database? A database is a place you keep information. What do you mean by information? Well, it's a lot of facts. What are facts? Well, facts are what logicians call propositions that happen to be true ones. So a relation is really nothing more than a collection of propositions. That's exactly what it is, and Ted had this brilliant insight that what we needed to do was store propositions in the database. And the way to do it is relations. And as I said this morning, if you realize that, if you think about a database that way, you are inexorably led to the position that the database has to be relational. Everything else is just implementation stuff. I don't think Ted ever described it that way, but I wish he had, because that's what it was.

Haigh: And so to wrap up with your time in England, you mentioned, I think, that you started the database management book in 1971 and that you wrote it in 1972.

Date: Yes.

Haigh: But yet, it didn't appear until 1975.

Date: Yes. That was a sorry story. I wrote it, and I'm very quick when I start writing. I wrote it very quickly, but I was in IBM. And for someone in IBM in those days to publish something, it had to go through the clearance procedures in IBM. And so I submitted it to the clearance procedure, and of course the problem with the book from an IBM point of view was that it was not coming out saying that IMS was the greatest product ever invented. So everybody who reviewed it in the clearance procedure did two things. First, they found something I had to change. And second, they found somebody else who had to review it. This process was clearly going to go on for a long time! At the same time, there was this thing called the System Programming Series that IBM was sponsoring. And I knew about this series of books, but I had made a conscious decision that I did not want my book to be in this series because it was an IBM series, and the book was my work, not IBM's work. And I wanted it to be separate, so I did actually hawk a proposal and sample chapters around to several publishers, and I got some proposed contracts, but I wasn't getting the clearance. What then happened was that Ted Codd, who was on the editorial board for the IBM series, took the manuscript to the rest of the board, and they said, "Ah, this is exactly the kind of book we want to publish in

the series." And then they got into a battle behind the scenes, which officially I didn't know about. They wanted to publish the book, but other portions of IBM did not want the book to be published at all and the fight went on for a while. Eventually, the editorial board threatened to resign if they could not publish this book.

Haigh: And I think we have the editorial board listed [inside the book], yes.

Date: That's probably true, yes. So there was a horrible battle, and there was blood on the floor.

Haigh: So were those people primarily in IBM research?

Date: In IBM anyway, but they weren't all in research. But they won, and so the book was finally cleared and published in that series, which was not my original intention, but that's the way it got there. And it was published in 1975, so it was sort of sitting in limbo for two years. It had been written in 1972, but it took two years to get through the clearance mechanism.

Haigh: Looking at the book, one of the distinctive features you've already mentioned is this division of the database world into three different groups: network, hierarchical, and relational.

Date: Yes.

Haigh: Now another thing is that it's very clearly written.

Date: Thank you.

Haigh: Obviously, it's nicely done. It has these exercises, and then it also has these quite detailed references and kind of an annotated bibliography.

Date: Yes. I'm glad you mentioned that. As I told you, I was reading all the research papers I could find, and I found for each one I had to write a précis for myself so I could understand what the contribution of the paper was. So when I came to doing the book, it occurred to me that those précis were very useful to me, so they probably would be useful to other people, too. So, unlike most other books I know, which just have the references, these are annotated references. And yes, I think that was a strong feature of the book, and it continues all through the other editions, by the way.

Haigh: And it seems, for an introductory textbook, to bring it closer than most of those books would go to the current research literature.

Date: Well, of course, I did have that agenda. I believed in relational. I wanted people to understand that. So in that sense, it was academic. It was talking about something that was not yet a product

Haigh: So that's an interesting choice then to put relational on the same basis as these other two categories that you're demarcating, even though if you look at the discussion of the actual systems, the relational chapter is talking about a number of prototypes. And it looks like almost all the citations were technical reports and so on.

Date: Well yes. Most of the published work, the academic work on database, was relational. I mean, what Ted did was put this subject on a solid scientific footing so that people could now investigate interesting things like the nature of data, and how do you do concurrency control, and how do you do security, and how do you implement it efficiently, and so on. So it's obvious that 90 some percent of academic work published after 1970 was based on the relational model. However, IBM made it very clear to me that I had to make it very clear in the book that relational was just a theory and there weren't any products. And IMS was a product; you'll see that over and over. So I took great joy a couple of editions later in taking out all of that stuff, because now there were relational products, and I could not be seen to be denigrating the competition.

Haigh: And then the tutorial section of the book gives in depth information on how to use IMS.

Date: Yes. I always felt, a little sneakily, that actually the IMS section was a very strong part of the book, because the IMS documentation from IBM was impenetrable. I'm blowing my own trumpet again, but I think it was a good tutorial on IMS, and people actually used it for that purpose. But it was also rather ironic considering the same people who were resisting the publication of the book turned around and started using it to teach IMS. It was funny. I remember I was at a conference in the U.K., an IBM user conference, a European Share meeting. And we were at the cocktail party the first evening, and a guy comes up and sort of insinuates himself. Like I'm talking to you, he stands here, and eventually we admit him to the group. And he turns to me and says, "You're Chris Date." I said, "Yes." He said, "You wrote that book." I said, "Yes." He said, "I tried to stop publication of that book." Great, now what am I supposed to say? But that was not an isolated incident. Several people told me similar things.

Haigh: And there's also some attempt there to explain the CODASYL standard. Was that something else that people inside IBM objected to?

Date: Yes. IBM was a little ambivalent about CODASYL. Basically, they didn't like it because there were already systems (I mean competitors to IMS) that supported something

very close to the CODASYL report. By the way, it was not a standard. It was a proposal but not a formal standard. Anyway, there were products out there already that were competitors to IMS. So for that reason, IBM did not like CODASYL. At the same time, IBM did have representation on the CODASYL Database Task Group, and so they were supporting it in a way. Sharon Codd was one of the representatives for a while. But I'm having difficulty remembering exactly. I know that there was a period of time when IBM would not allow IBM employees to talk about CODASYL outside the company. That regime was in existence for a year or so, and I suffered from that one, too.

Haigh: And then, when the book was finally published, did you have a sense of who adopted it and how it was used and what the responses were from the readers?

Date: Well, the responses were very good, and the book was fairly quickly adopted by colleges and universities and became…well, I won't say a standard, but a very widely used book in universities. The book appeared in February of 1975. In May of 1975, there was a National Computer Conference in Anaheim. And I gave a tutorial there on this new relational stuff, and associated with the conference there was a sort of trade show. Addison-Wesley, the publisher of the IBM series of books, had a stand there. And I went by afterwards to see what was happening. And the guy there said, "I'm sick of hearing your name." He said, <laughing> "I just sold 1,800 of your books in the last hour and a half." And then, of course, I got reviews, and the reviews were, though I say it myself, they were good. They said it was clear and it was balanced, believe it or not, and understandable.

Haigh: So I think then we should probably return to your career, and I think the main thing we have to talk about before you moved to America would be the famous ACM debate.

The ACM Debate

Date: Yes. I'm glad you've come to that. I think Ted and I had corresponded for a couple of years, and I forget exactly how this happened, but I went to the Share Conference in Montreal, and this would have been 1973. Ted was also there. And it wasn't a regular Share meeting; it was a special meeting on database, and I have the proceedings of that at home, as a matter of fact. Anyway, the highlight of the conference was Charlie Bachman, who had just been given the ACM Turing Award. Charlie gave a dry run of his Turing lecture, a famous talk called "The Programmer as Navigator." And I was sitting next to Ted in the audience, and I could feel Ted gripping his chair, and the instant that Charlie sat down, Ted leapt up. And he was the first to the microphone. And he said of course, "I'd like to congratulate Charlie on the Turing Award. That's really great, and he deserves it; he's a very nice guy," and all that sort of thing. Then he says, "None of that affects the fact, however, that what he is saying in his paper is dead wrong." That was typical Ted! Then he said, "There are so many things that need to be said in order to deconstruct Charlie's argument. But it would not be appropriate to do now. This

is not the right forum to do it, standing here at the microphone. So instead, I would like to issue a challenge that Charlie and whoever else he wants meet with me, and I'll have some support, at the next Share meeting. And we'll have a formal debate on these matters." Now, Ted and I had been talking ahead of time about this paper of Charlie's, and I think it was actually my idea. I said, "This is what we should do. Instead of getting into a shouting match here, let's do it properly in a controlled environment." So Ted proposed that, and it was accepted.

It changed from being the next Share meeting to the SIGMOD Meeting in Ann Arbor, Michigan in May of 1974. And on Charlie's side, there was Charlie Bachman himself and Ed Sibley, who I think was at the University of Maryland – I'm not certain of that – and a guy named Jim Lucking, from ICL in England, because ICL had signed onto the CODASYL stuff at that time. And on Ted's side, Ted was the principal speaker. I was supposed to be his backup, but I had visa problems, and I was stuck in England trying to get to California at the time. So Dennis Tsichritzis stood in for me. But the proceedings exist, and all these names are there. I have them myself, but you can get them through the ACM. Dennis was a professor at the University of Toronto, and he gave my presentation. And the third person was Kevin Whitney. Kevin, I believe, died a few years later, but he had built a small relational system, so he was the third speaker. And I should also have said that Ted and I had prepared very carefully for this, and we actually wrote two papers that appeared in the proceedings. Ted wrote one which was about the differences between the approaches from the end user's point of view. And I wrote one on the differences from the application programmer's point of view. Both papers have both names on them, but the fact is Ted wrote one, and I wrote the other. And reading through the proceedings, I don't know if this is true, but it certainly appears that Charlie and his people had not prepared as carefully as we had. Their arguments did not frankly seem that coherent. And I think it is true to say that after the debate, the majority of the people in the audience thought that Ted was on the right side. Of course, the audience was mostly academic, so they probably would think that, but that was the outcome. I wish I had been there.

Haigh: So that was the famous debate?

Date: Yes.

Haigh: Now, I think at least in retrospect, it seems that the argument in favor of the network and hierarchical systems would mostly be one of operational efficiency dealing with high transaction volumes.

Date: Well, they were certainly tailored to do that. And of course, they had running code and they had an installed base. Relational stuff didn't even really exist yet in product form.

Haigh: Yes. And the argument in terms of relational systems would be flexibility, ad hoc querying and to some extent, speed of development.

Date: Well, there are lots of academic arguments which maybe we shouldn't get into right now. But things like completeness, knowing what the capabilities of the system are without having to scratch your head and figure out whether or not something can be done. But I think one of the overriding concerns in Ted's mind was a pragmatic concern, as opposed to one of academic elegance and theory and so on. It was that, in the future, the vast majority of users of computers were going to be computer naïve people: people who didn't understand internals and shouldn't have to understand them. And so for them he said, what is the most elementary and intuitively understandable way of representing information? Surely, it is just tables. We use tables all the time, for all kinds of things. And I had observed myself, talking to people about their database requirements, and asking, "What's your data look like?" They will start drawing tables, even if they're IMS users. Tables are a very natural structure. And so we start with that elementary idea. I sometimes characterize the database engine as a sort of cut and paste machine for tables. If you imagine you have all these tables of information, then you can cut bits out here or cut bits out there, paste them together to join tables, and so on. And all the information you want can be obtained by a series of cut and paste operations like that. And that's a metaphor that people find very easy to latch onto. So, ease of use for the end user was a big driving force – getting away from the sordid details of how things are really done on the machine. Users should not have to bother with that stuff. And if you achieve that separation, now you have this wonderful freedom. You can change things on the machine. You can port to a different system. You can replace the operating system. You can do all kinds of things because it doesn't change the interface for the user.

Haigh: So is this concept of data independence something that had been formulated at the time of the debate?

Date: Oh, yes. I couldn't say when the term first came into being, but it was certainly before 1970, so late 1960s. It's not a very good term, by the way. It doesn't capture what it's really all about. But in Ted's 1970 paper, he says that one of the big problems with the hierarchical and network approaches is they give very little data independence. We need this sharp separation. And it's in the original paper, although I believe what I said about the end user and so on, ease of use for the end user, that was a big driving force in his mind, I know, and he wrote it down later. But in that paper, he focuses very much on data independence as a goal.

Haigh: And I know you weren't there at the meeting, but in the couple of years afterwards, was your sense that the meeting changed anyone's mind, that it had a direct influence as opposed just to being a kind of symbol of the way things were going anyway?

Date: Very hard to say. I don't know.

Haigh: Well, then, returning to your career, I understand that the reason that you couldn't get your visa was that you were already in the process of getting the paperwork to move to America.

Date: Yes.

Haigh: So how did that come about?

Date: IBM at that time had a grand project called FS, Future System. This was going to be the all-singing, all-dancing, replace-everything-out-there system, and it was a massive project, billions of dollars. And they were, as it were, scouring IBM locations around the world looking for people with specialties in various areas and bringing them in. So there were 12 people who went from Hursley to California, or rather 12 families. We were one of them. And my particular job within this grand scheme – they were going to build a brand-new database system – and my job was to define database extensions to the high-level languages. Not just PL/I but all of the then-important ones – COBOL, FORTRAN and PL/I. The approach depends on your point of view. Either we needed language extensions to support the database system or we needed a database system to support the languages. It depends which way you look at it. But that was my job. And, of course, I proposed the same thing, that we should have relational extensions. And I actually defined a set of relational extensions, defined them abstractly, and then went through a separate process of mapping them to the languages. I was then informed that we weren't going to do that, because this all-singing, all-dancing database system was going to support not just relations but also hierarchies for the existing IMS base and also networks, so we could attract CODASYL customers. We were going to do all of them. I thought this was the most terrible idea. But we had a task force, that's the way IBM used to operate in those days, a task force basically to tell me how to do my job. Don Haderle was on that task force, by the way. And I was instructed to go off and come up with a way of doing all three in the various languages.

The Relational Model

Haigh: Now, I think to some extent the gist of some of the comments you've made is that the relational model captures the fundamental essence of data storage. So why didn't you view this as a chance to show that the hierarchical and network things can be broken down into their constituent relational operations underneath?

Date: That's a sort of softball question, or at least a leading question, because that's in effect what I did. All I wanted to do was have a relational interface. That's what I wanted my users to see. But I was told, "No, no, no. We have to have hierarchies and networks as well." So it occurred to me that if you think of a network, you have records and you have links between them. And the links can be many to many, in effect. And a hierarchy is just a special case of

that, where the links are limited to being one to many. So you could say, at least to a first approximation, if you have a clean idea of what a network is and a clean idea of what a hierarchy is (as opposed to the rather messy ideas in CODASYL and IMS), you could say that all hierarchies are really networks. Hierarchies are a special case of networks. So everything you can do with a hierarchy you can do with a network. And then it occurred to me that if you have a hierarchy with records and links, but there are no links, you have just records, and that's relations. It's a good first approximation. So you can say all relations are hierarchies, but some hierarchies are not relations. All hierarchies are networks, but some networks are not hierarchies. And if you recognize this pattern, you can then design the language to reflect the pattern, and that's exactly what I did. I had a core of functionality that you needed for relations, and therefore you needed it for hierarchies as well because relations are hierarchies. And therefore you need it for networks as well. But then there's another layer of functionality around the core which you needed for hierarchies but not for relations. And then another layer again that you needed for networks and not for hierarchies. So the proposal had this onion-layer structure. And the message was that you can do everything you want in the core, the relational core. These other layers add only complexity. They don't really add functionality. I said functionality but I shouldn't. They add complexity. There are new operators you need, and other new things. But there's nothing useful you can do in the outer layers that you can't do in the core. So my purpose in designing this language was to spread that message, to say, "Look, you know, you can do it all here. You don't need the other stuff."

Haigh: And were you still in a situation where you were imagining that the database engine and the optimization and everything would be done by some other group somewhere else?

Date: Oh, definitely.

Haigh: So you never had to think about what that would mean from an implementation level?

Date: Well, over in the Data Management department, they were already building the implementation. But just to finish up, I designed the language with a particular agenda. I wanted to use it as a message, as a way of spreading my message. What happened, of course, is people got hold totally of the wrong end of the stick and said, "This is wonderful. We can do all three in the languages." So they started seriously trying to implement this whole thing. I called it UDL, for unified data language. They tried to do the whole thing, but that was never my intention. I wanted to use it to argue you shouldn't do the whole thing.

Haigh: As I understand it, if the thing would have worked right, wouldn't the complexity of the other two models have been embedded in the language so that the things that came out of the bottom of the compiler, for the engine to implement, would have been relational operations?

Date: Probably not, no. To get an efficient implementation under the covers there'd have to be ways of bypassing the relational core. Somehow, if the user saw a network structure, it would map to a network structure. This of course, is precisely the problem with that stuff. It is the physical level exposed to the user.

Haigh: Because in those days query optimization wouldn't have been able to cope with it?

Date: It would have been a challenge. But the message was that you don't need the other layers. They got hold of completely the wrong end of the stick and started actually trying to do the whole thing.

Haigh: So you were being too subtle?

Date: Maybe. And then what happened is that I was originally on assignment for FS, as I told you. And the FS project was killed and IBM – somebody once said, "We're spending billions of dollars on this thing, but no one knows what a program is in FS, no one knows what a job is, no one knows what a task is, you know? It's completely hopeless." Eventually they saw the light; cancelled the whole thing. Most of the assignees went home. I did not, because although FS was killed, something was going to come along and there were still going to be database requirements and the languages still had to do something, so my job still existed. And in fact by that time I'd become very much enamored of California and decided I wanted to stay in California and so I transferred from IBM U.K. to IBM U.S.

Haigh: At the risk of asking an obvious question, what did you like about California?

Date: Can I stack that for a moment? Let’s finish up the other thing first. So the okay had come through. I went back to the U.K. to sell the house and all that sort of stuff. And while I was away, there was a massive reorganization back in the U.S. such that the languages people and the database people were no longer talking directly to each other. They were very separate pieces of the company, and they met only at a very high, very senior level. And the languages people were saying, "We want to do Chris's UDL." And the database people were saying, "You can do that, if you like, but we're going to do SQL." And that's what happened, and I was literally out of the loop at the time, so I had no input to that decision. Which, by the way, could be interpreted to mean that when I criticize SQL, it could be viewed as sour grapes. Absolutely not. I didn't want to implement my language anyway. My problems with SQL are much deeper.

But to go back to the other question, why did I like California, there are lots of reasons why I wanted to stay. Professional reasons: I'd been working on database stuff in the U.K. for three or four years and I knew all the names of these people who were writing papers, but suddenly I

was meeting them. They were physically here, or at least at the end of a telephone. (In those days, even the idea of making a telephone call from the U.K. to the U.S. was a big deal.) It was exciting. And at that time, San Jose, California was the center of the database world. If you wanted to do database stuff, this was the place to be. So there was that. And then there were lots of other personal things. I, like most Europeans when they come to California, they're impressed by the space. It's huge compared to how things are cramped in the U.K. I love the climate. I love the wildlife and so on, and there does seem to be so much more and it's more easily visible here. I liked the people I met tremendously. And I had very personal reasons too, to do with health. I was allergic to the British climate and I didn't realize that until I lived here for a few months and realized that oh, yes, flowers have a scent. I mean, I knew this intellectually but I hadn't experienced it. So that was a very personal thing. But there were many reasons, so eventually I made the decision to switch. That's not to say there aren't some regrets. You know, I miss some things about the U.K., of course. But on balance, I like it here. I love the country here. The political scene is another matter.

SQL

Haigh: All right. Now to get back to some of the comments you made previously, so you went away, you came back and they decided to do SQL. So this is the first time that SQL and presumably the associated System R project have impinged on this story.

Date: Yes, well though, of course, I was in touch with the System R people all the time. I never worked on System R. They were in Research; I was in development.

Haigh: Where were they physically? Were they in this area, too?

Date: Yes, San Jose; the San Jose Research Lab, which was at Monterey and Cottle Roads, San Jose, I think, at that time. And I was in Palo Alto and then later, in Santa Teresa.

Haigh: And were they in touch with Ted Codd?

Date: Well, he was in Research also. But I have to say I don't think there was all that much communication between them. I've often wished that Ted had been the manager for that project because I think we might have avoided some of the mistakes. But that's water under the bridge. Well, backing up a bit, I mentioned in the symposium this week how he had been actively lobbying for getting a relational project going in IBM, this thing about gamma zero, gamma one and so on. One of the interfaces in this thing was gamma four, which was to be a natural language interface. And all through 1972 and 1973, he was working on implementing such a thing. He built a prototype called Rendezvous which was support for English-language queries on a relational backend.

Haigh: English language in the COBOL sense?

Date: No, real English language. Natural language. And he built a prototype and it was very robust. It worked very slowly, but it worked very well. He wrote it in APL. You could enter queries like how many parts did supplier Jones supply to projects in Los Angeles in May, 1973, literally in those words. And what the system would do is it would take little fragments of this text – like parts, I know what parts is, that's the parts relation, and how many, that means count and so on. But other bits it wouldn't understand. Say Los Angeles, it wouldn't understand, so then it would come back and say, "When you say Los Angeles, is this a property of a part or a project or a supplier?" And you say, it's a property of a part. Okay. So is it a part number or a part weight or a part color or a part location? It was very tedious, you see, but it engaged the user in dialogue and gradually built up a complete relational expression. And, of course, because it was a formal system, the relational model, it knew when the expression was complete and then it would go to the back end and do the query and come back and give the answer, again in English. It was elegant. It was a nice idea, you know? But in a way, possibly a little bit of a dead end. It would have been more fruitful, I think, if his energies had been devoted more toward getting that backend system into shape. But no, he didn't do that.

Haigh: Oh, by the way. In terms of the development of the terminology, when did people first start talking about tables existing in relational systems (as opposed to relations)?

Date: Well, SQL certainly did that. But in his earliest papers Ted pointed out that the abstract mathematical construct of a relation could be conveniently represented on paper as a table, and this was one of the attractions. It has a very nice on-paper representation, which makes it easy for people to understand.

Haigh: And presumably when you're talking about a table, rows and columns follow naturally from that.

Date: Exactly. The trouble is, though, we were caught in a dilemma, really. See, a table and a relation are not really the same thing. And an attribute and a column are not really the same thing. And a tuple and a row are not really the same thing. They're close, but they're not the same. One of the mantras I always like to appeal to, Wittgenstein once said, "All logical differences are big differences." And there's a logical difference between a table and a relation. So we ought to have bitten the bullet and used the formal terms from the beginning, relation, tuple and attribute. But nobody would ever have listened to us. You had to make the message more palatable. So we talked often in terms of tables, rows, and columns, but with hindsight, I think it was a mistake because there's baggage attached to those words. People have intuitive expectations which are not necessarily fulfilled.

Haigh: Like order?

Date: Things like that, yes. Well, it was a big mistake in SQL. A table has a left-to-right order to its columns, but a relation does not have a left-to-right order to its attributes. SQL has a left-to-right order to the columns of a table. It sounds like a trivial little thing, but it causes immense complexity. That's one of the many, many reasons why I keep saying SQL's not really relational.

Haigh: Neither you nor Codd was directly involved in the System R effort.

Date: Right.

Haigh: When did you first see SQL?

Date: Well, it was before 1974. As Don Chamberlin said, the first paper external to IBM was in the 1974 SIGMOD proceedings. And I'd certainly seen it before then. I don't remember the exact date.

Haigh: And do you remember what your initial reactions were? Did you want to run over and start talking?

Date: No. It's really very mean of me to complain so heavily now because I didn't complain heavily at the beginning. I did complain when the decision was made that SQL was going to be the language; at that time I had serious complaints. But in the first days I didn't. I've already mentioned that in some respects it was a little similar to what we had proposed with PL/I. But you see, Ted had relational algebra and relational calculus. The SQL guys said, "Those are both what you might call user-hostile. We want something else. We want a third way." And they came up with what became SQL. Now, there's nothing wrong in coming up with a third way, provided you can prove that this third way is logically equivalent to the other two. That was not done, and in fact could not be done because it isn't equivalent. And so you had horrible things. I don't want to get too technical here but, for example, in the early days, you couldn't do a group by on a table that was already grouped. So you lost closure and things like that. So there were these problems. But, to be honest, I don't think either Ted or I really saw the depth of these problems in the early days.

Haigh: Right.

Date: I wish we had, you know? We saw problems, but not the deepest problems.

Haigh: And were those limitations coming from mutations of the underlying database technology or were they just introduced by carelessness in the language definition?

Date: The latter, I believe.

Haigh: Okay. Now, just to get the timeframe on this, have you given a date yet for when this decision to go with SQL was made?

Date: Well, it must have been 1977 because that was the year I transferred.

Haigh: And I think you told me previously that the whole software group that you were in moved to the new Santa Teresa lab?

Date: Yes.

Haigh: Now, as you'd said, you'd originally been there as a temporary assignee. The job had become permanent. So what was your job title? I mean, what was your official job description?

Date: You mean after the UDL stuff?

Haigh: Yes.

Date: The UDL stuff continued for a little while and, you know, prototypes were built.

Haigh: So through 1977, UDL was your full-time responsibility?

Date: I have a hard time putting exact dates on it, but yes, probably into 1978 or so. And the languages people were still serious about doing UDL and we even went to some outsource contractors to see if we could hire them to do this job. I remember we went to some company in San Diego. But at some point, somebody realized this was a complete waste of time. SQL was going to be it. And so I forget the exact sequence of events after that, but I know that one of the last jobs I had before I left IBM was working for Jim Strickland, who was here this week, and the title of the department was DB2 Technical Planning. So we were supposed to be defining technical requirements for various releases of DB2 and, to some extent, for SQL/DS, the lower-level product, because originally its home was also Santa Teresa. Later it moved to Endicott. But some of the decisions we were making, although they were for DB2, fed into SQL/DS as well.

Haigh: So, you moved to Santa Teresa in 1977 and you left IBM in 1983.

Date: Yes.

Haigh: That's a six-year period.

Date: Yes.

Haigh: And you were working full-time on database things through that whole time?

Date: It was always database, but I'm having a very hard time remembering the specifics, possibly because I wasn't enjoying myself very much. I really don't recall all the things I did in there. And, of course, I was actually writing a lot of books and doing things like that.

Haigh: We'll talk some more in a second about how the books were progressing. So why weren't you enjoying things so much?

Date: It's speculation on my part, looking back. I don't remember. But the very fact that I don't remember is probably indicative of the fact that I wasn't enjoying myself too much. Although it occurs to me, of course, one thing I was doing increasingly was going to user groups and the like and talking about this new relational stuff, which was now beginning to happen, you know? So I was doing quite a lot of that, both in the U.S. and overseas.

Haigh: Were you learning more from the users about what their actual needs were in the kinds of work they were doing?

Date: Yes. I was actually for a while the IBM rep to the Database Language Project within Guide. So yes, they would come along with real application problems and in the UDL days I'd try them out in my language to make sure it would be able to handle what they wanted to do and that sort of thing. So there was a lot of user input to that, actually. That started almost as soon as I arrived in California. I went to one of these meetings about a week after I arrived, in Anaheim. That was a funny one. This languages project was trying to figure out what the languages should do from a user perspective, what they should do for database stuff. And they'd set themselves five sample problems. And they went away, and each one coded the five problems in his or her favorite system: Total, IDMS, IDS, whatever, and of course IMS. And then they came back and presented their solutions, and I had been brought in just for the afternoon to talk about this new relational stuff, just to tell them what it was. And so I saw these five problems cold and I was able to show them that each one was just one line in relational calculus on the blackboard. They were impressed by that. And the then-IBM rep said, "Let's have a coffee break." He took me outside. He said, "Look, I don't want you to come on too strong about this relational stuff." And I said, "Why not?" He said, "Because we don't have a product." I said, "Well, maybe we should get a product." It was only seven years after that that SQL/DS was announced.

Haigh: Right. And you had some involvement with SQL/DS and DB2. Were you helping them get the requirements straight? What was your role?

Date: Well, I think the honest truth is my influence on the design of those products was absolutely zero. Everything I proposed – like support for foreign keys, something we heard quite a bit about this week – it did not get into first release. I'm not sure which release it did get into, but things that I thought were really important for users as well as being integral parts of the relational model just didn't get there.

Haigh: Support for foreign keys in the sense of enforcing referential integrity?

Date: Yes. As Ken Jacobs kept saying, "Allowing the user to state the requirements declaratively and have the system do the work." And they didn't do that in the first release. And of course, the problem with languages – and after all, we are talking about a language interface – the problem is you have to get it right the first time, because any mistakes you make in the first release, you have to live with forever. So if you allow, for example, a table to exist without a key, forever you have to allow tables to exist without keys, which means you have to allow duplicate rows, which leads to all kinds of problems.

Haigh: So if you don't have anything else to say about what you did at IBM, perhaps we should catch up with the books you had been working on during that period.

Date: I'd rather talk about that, yes.

Evolution of the *Introduction to Database Systems* Book

Haigh: Okay. So, let me see. Looking at the dates here, I see there have been so far a total of eight editions of the *Introduction to Database Systems*.

Date: Yes.

Haigh: So how did that book evolve over its first 10 years?

Date: Well, you recall that the first edition was published very late; written in 1972 but not published until 1975. Consequently, by the time it appeared it was already out of date in a sense.

Haigh: Yes.

Date: So I immediately started working on a second edition and that appeared in 1977, so there was a very narrow gap there for that reason. It was really a five-year gap, but it looked like a two-year gap. And it evolved because – well, to be blunt, my own understanding increased, you know? I didn't know very much when I wrote the first edition, particularly in the relational field. Now I understood more about relational. And, of course, new developments were happening in the academic world, which were really interesting. New optimization techniques were being invented, new ways of doing joins, design theory, dependency theory was being developed, and the importance of integrity constraints was beginning to be understood. All these things fed into my brain and, in turn, I wanted to write them down. And the horrible, pragmatic thing was the book was being used as a university textbook. And the publisher sort of lives on a five-year cycle, you know? If the book gets to be five years out of date, it's obsolete. So every five years, if you have a contractual obligation, they're going to come back and say, "We need a new edition." That has happened every five years, really, ever since, right up to the eighth edition.

But I've learned more and more about the subject. The book has grown. I'm sorry to say it's about 1,000 pages now. But so are all the other books, not just mine. One thing that happened, though, was that this book was intended really to be an introduction and there are a lot of other topics, like recovery and concurrency, that there wasn't really room for in detail. There's a chapter, but not much. So I had the idea of writing another book on more advanced aspects of database stuff. And I had eight chapters – recovery, concurrency, I think security, and so on. And I didn't know what the hell to call this book. If I called it advanced, it would put people off. So eventually I called it Volume 2 and called the first one Volume 1, retrospectively. That caused immense confusion. People never understood that. But over the years, all the material in Volume 2 gradually found its way back into Volume 1, which is one of the reasons Volume 1 is now so much bigger than it was originally.

Haigh: And during this period, other database texts must have appeared.

Date: Certainly.

Haigh: Do you feel there was something distinctive about yours that gave it a special niche? Was there anything that you had to change to keep it competitive?

Date: There were differences. Whether they are regarded as pluses or minuses is not really for me to say, but there were certainly differences. And again, I'm blowing my own trumpet and I apologize, but I do believe the treatment of the relational model in my book is stronger than any of the others out there. And there are quite a few topics that I talk about in some depth that other books don't, like missing information and type theory and inheritance and various things. On the other hand, other books have things that my books are weak on. There are certainly books out there that are much stronger on implementation techniques, for

example. My interest was always in getting the model right first, and worrying about implementation afterward, which is exactly what Ted did, too. And it shows in the books. There is material on implementation but not as much as in some of the competition.

Haigh: And presumably the amount of material on the network and hierarchical systems would start to diminish at some point?

Date: Correct. It gradually atrophied. It went down to becoming just one chapter on each from being several chapters in the first edition, and then it migrated to becoming an appendix on each. And then it disappeared entirely.

Haigh: Did you include material on data modeling techniques?

Date: Yes, one long chapter in the current edition.

Haigh: So, let me see. Also, in 1983, *Database: A Primer*.

Date: Yes. That was always intended to be a low-entry kind of thing. It was meant to be a paperback from the word go. It was written at the time that PCs were just beginning to be ubiquitous. There were some little database systems that ran on PCs. So the idea was to explain just the elements of the stuff at a very basic level. And I flatter myself, but I think it was a good book of its kind. It's out of print now, but it was a nice little thing at the time.

Leaving IBM

Haigh: Okay. So I think that brings us up to your departure from IBM.

Date: Friday, the 13th of May, 2:30 in the afternoon.

Haigh: And how did that come about?

Date: Well, it had to do with a number of things. One was, as I told you, I was doing a lot of presentations outside. I was getting a phenomenal number of requests for external presentations, and I turned down 60, 70 percent out of hand. I simply couldn't do them for time reasons. And then every presentation I did, I'd have to get clearance from my management and they had to have the freedom to say no, so I'd always ask for some that I didn't want to do as well as the ones that I wanted to do. So I did maybe 10, 15 percent of the ones that got requested. But I slowly began to realize I could do this 100 percent of my time and do it for myself instead of doing it for IBM. That was one thing. And I found the IBM environment increasingly frustrating in various ways, you know? So I snuck around a little bit making sure

that there would be some consulting work for me if I left IBM, and there was. So I resigned and worked for myself. And that was on a Friday. I took Saturday off. On Sunday, I taught myself to use a word processor. On Monday, I started writing the book on DB2. That took a week.

Haigh: So what kind of consulting work did you have in mind?

Date: Well, I'm not a consultant in the conventional sense. I do not go and help people fix performance problems or help them design their databases or anything. What I realized was that much of consulting is really education, and I'm good at that. I think I'm good at teaching because I'm a very slow learner and therefore I can understand the difficulties that people have, and I found that I was a good teacher. So to the extent that was true, I was consulting with many companies but it was mostly going in and giving presentations. And that included all the major database vendors, including IBM. My first consulting job was back with IBM, where I had worked. I went in to talk about the advantages of primary and foreign keys. There were two differences. I was doing exactly what I had been doing when I worked there, but with two differences. One was they were now paying me much more than they paid me in IBM. And the other was, they were listening to what I was saying. You know the definition of a consultant is someone from out of town. I was telling them the same story I'd been telling them before, but now they were paying attention.

Haigh: Were the audience for these talks mostly people who would be involved with designing database applications and needed this kind of grounding? Or were you talking to research groups about the latest conceptual developments; or were you doing a mixture?

Date: All of the above. In the early days, it was sort of one-off things. But I gradually developed some proper seminars, like week-long seminars, and taught these seminars at a variety of locations through a number of different organizations. And they would be essentially open to anyone who paid the money to come. But the audience, I would say, would be 50 percent DBAs, database administrators, and most of the rest were probably database application designers or application programmers, with a smattering of folks from DBMS implementers and standards groups and academics.

Haigh: So that would be covering the same kind of material that you have in the textbooks, then?

Date: Yes and other material that was not in books yet.

Haigh: Yes.

The Codd and Date Companies

Date: So that started in 1983. And Ted also left IBM about then. I don't exactly remember when that was, but I want to say 1984. And for a short time, we were both out there being independent consultants. And Sharon Codd came along and said, "This is stupid. You're competing against each other. Why don't you get together?"

Haigh: Right. Why did Ted leave?

Date: Some of the same reasons that I did – frustration with the status quo and the realization that he could probably do better financially, going around being the relational expert.

Haigh: And was he looking to do the same kind of work that you were doing?

Date: Well, he certainly did do some lectures. But to be frank, that was not his forte. He did more consulting with vendors, advising them on the design of their products or assessing their products or their product plans, that sort of thing, which I did some of but not very much. There must have been other things, but I'm blanking on it for the moment. But anyway, we were in the same space, as the saying is, and Sharon said that we should get together and we did. And we formed various companies.

Haigh: And Sharon, was she married to Ted at that point or not?

Date: No. No. Her name was Sharon Weinberg at the time. And so the three of us became partners in these companies. And I agreed it made sense; on paper, it made sense. At the same time, I was very wary about getting back into a company, having just escaped from a company. And so I said at the time, "Well, okay, it does make sense and I'll do this for three years, but after that I think I'll probably leave again." So I was upfront about that. It turned out to be five years, but eventually I did leave and go back to being independent.

Haigh: So you never had the idea that this was a company that you wanted to grow and stay with?

Date: Well, of course we did, somewhat. We wanted to grow it to the point where it was worth a lot of money and then sell it to somebody, you know? That was the exit strategy. And we did have some people who were interested, but it was dragging on too long.

Haigh: Did you bring in additional employees?

Date: We certainly had an office staff and we had a band of independent consultants that we would hire on a contract basis. One of our primary pieces of business was seminars. We ran lots of seminars and we had lots of people doing that, because, of course our own skills

did not cover the universe. So we brought in other people who were experts on database design, for example, to teach that kind of stuff. And, for a while we had, oh, I forget – I mean, the numbers kept changing – but certainly 15 or 20 people around. We opened up a subsidiary in England. Actually, it was a European subsidiary headquartered in England. We had another subsidiary in Italy. So we were expanding like crazy for a while. But we made every mistake in the book. We had no idea how to run a company.

Haigh: What kinds of mistakes?

Date: Well, for a start we wanted to fund it ourselves. We did not want to be beholden to some other investors so we never got venture capital or anything like that. That was a huge mistake. And we didn't have the knowledge to run a company, and we didn't realize we didn't have the knowledge. Ted and I had no interest. We left it to Sharon. She was the CEO, in effect, and Ted and I just carried on doing our thing. And Sharon didn't know how to run a company either, and it took us all a while to realize that that was the case, herself included. She would agree.

Haigh: What was her background?

Date: Degree in mathematics. She got into database; she was in IBM also. I first met Sharon myself on the trip when I was invited by Ted to tour around the country presenting my ideas from Hursley. Sharon was in New York and that was one of the places I presented, so I met her there. That was 1971, I think. Then when we all came out to California, Sharon also moved from New York to California and she was in the languages department – basically, database languages. So, we were professionally connected for quite a while. As she mentioned this morning, she later became a presenter in the customer briefing center in IBM. So part of her job was to tell customers about this relational stuff. She was a good presenter. And the three of us had been friends for quite a long time. And then we formed the company and later, Ted and Sharon were married.

Haigh: Interesting. So did the fact that two of the three founders were now married to each other make any awkwardness for the company?

Date: Well, I thought it might, which was a contributing factor to my leaving. But since I was going to leave anyway, I thought maybe now was the time to do it. I don't think we ever had a problem, but we certainly could have had for that reason. It's kind of a strange situation, isn't it?

Haigh: So you left and carried on doing pretty much the same stuff you had been doing all along then?

Date: Johnny One Note, as Mike Blasgen called me this morning. But you see, the beautiful thing, or one of the beautiful things about this material is that it has infinite depth. I'm still learning new things all the time. Even though the basic ideas are so simple and you can explain them in five minutes, the depth is incredible. So, I'm still learning myself. If I was not learning myself, I would have given up a long time ago. That's a fact. But, you know, there's still new stuff all the time. And I have a great colleague in the U.K., Hugh Darwen. He used to be at IBM U.K. but has now retired. Incidentally, he was the IBM U.K. rep to the International SQL Standards Committee for many years, and he kept them a bit more honest than they might have been otherwise. But anyway, Hugh and I have been working for quite a long time on something we call *The Third Manifesto*. There's a book on that in the list. And I could talk for a long time about that, if you're interested. But we're still doing it, so that work is continuing.

Haigh: Okay, so what that leaves us to talk about is your career as it's developed since 1991, feelings on changes in the field, these technical books. I think we have time to review them. You have a number of them there, so you can choose which ones you think are worth discussing.

Additional Books

Date: Well, I can sort of summarize. The next one on the list after the ones we already mentioned was *A Guide to the SQL Standard*, and the background to that one I remember very clearly. I was on a plane flying from Rio de Janeiro to Los Angeles, one of those horrible night flights. And it was clear that the first SQL standard, what became SQL/86, was going to be ratified in a few weeks time. And I remember thinking, "Well, somebody ought to write a book about that, because the standard itself is very hard to understand. Somebody should write a textbook." And I thought, "Well, I could do that, but I don't want to do that. It is not intellectually satisfying. It's not something I want to do. I'll let somebody else do it." And then I thought, "But they won't do it right!" So, I got back and it was a weekend. I called my publisher at the time, Addison-Wesley, on Monday morning, said, "I've got this idea for a book." I explained it. They said, "Fine. Go for it. We'll send you a contract." And I wrote the book and it was done by Wednesday, because it was all in my head. The structure, everything was laid out. And it was only a little skinny book, but it was a good one of its kind. I flatter myself, but it was good. And that went through four editions. The last edition talked about SQL:1992, which is an immensely complex document, but by that time I'd brought Hugh Darwen in to help me and we did that book together. So, I would be reading the specs, come up with questions, feed them to Hugh, who would take them back to the SQL Standards Committee, and they would say, "Oh, my God, Chris is right. We've got to change this, you know?" That sort of process went on. And again, it was not a satisfying book to write, but it was a good one to do. To the extent that – again I'm bragging and I apologize – but when I want to look up something about the SQL standard, I look at our book, not at the standard because the standard is almost impossible to read. One reason is that the structure is so poor. The great thing we did in our book was, for example, we had a chapter on security. And every little thing about security, which is scattered through 2,000

pages in the standard, we brought together in one chapter. Same for integrity, same for table expressions, and so on. So it's a more logical structure. It was hard work, and I'm never going to do it again. That's why there's no fifth edition.

Haigh: What's your feeling about the impact that the various standards have had on the products actually offered by database manufacturers?

Date: Oh, it's been huge. In the early days, as Ken Jacobs from Oracle was saying, there was even a certification testing suite, you know. And so products really had to pass that test in order to claim legitimately that they complied. There is no test now, so some of the claims might be a little suspect. In fact, I believe they are because SQL:1992 – never mind the current standard, which is SQL:2003, I think – was full of holes and contradictions and inconsistencies. It was not implementable. No one could implement it. There are problems. We documented many of those problems in our book in Appendix D. And they have not, to my knowledge, been fixed, most of them. So there's a little bit of a confidence trick, you know. People are out there saying there's a standard, but it's not an implementable standard. But nevertheless, it had a huge impact in the early days. There's no question about that.

Haigh: Mentioning, as well as the books, I'm also seeing some papers. Have all your papers been in the same kind of synthesis mode, or have you published papers that you would say have an original research contribution?

Date: I would say that's been the case more recently. I always used to say in the early days that Ted was the inventor and I was the explainer, and that was our relationship for many years. But latterly, certainly since I left the Codd and Date companies, I've been doing things myself that I would regard as research on various things like integrity constraints, view updating, and so on. So some of the more recent papers I've published I regard as original contributions. But the vast majority of the early ones were tutorials. I'm always interested in explaining things to people. Many of the papers have titles like, "What is a Domain?" And I explain exactly what a domain is. "What is a Relation?" And so on.

Haigh: Do you want to say anything about the original contributions?

Date: Well, I'll just mention this one by way of an example. Do you know what normalization is about?

Haigh: Yes.

Date: I came up with a conjecture; if it were true and if I could prove it were true, it'd be a theorem. And as I said, I used to be a mathematician. I tried to prove it, and I sort of did prove it to my own satisfaction. But then I sent the proof off to Ron Fagin. Ron Fagin is the guru

with respect to normalization (and a lot of other things as well, by the way). And I asked him to look at it. What he said was, "This is really good. Your conjecture is true and I've proved it." [My own “proof” was nothing like correct, as it turned out.] So I have a theorem and we wrote a joint paper about it. I was very proud of that because that was the first original piece of work that I'd done, in a sense. But since then, there have been a bunch of other things. Many of them have been collected in some of these books, partly because I find it's an easier forum to publish in, to be honest. And then of course there’s the whole business of *The Third Manifesto*, which is a joint effort with Hugh.

Haigh: We should just get the citation.

Date: This is the original paper by myself and Ron Fagin. "Simple Conditions for Guaranteeing Higher Normal Forms in Relational Databases,” *ACM Transactions on Database Systems 17,* Number 3.

Haigh: Okay. And as your career developed and you published more and more books and gave more and more talks, was there a point that led to any kind of significant shift in what you were doing or how people treated you or your sense of what your career was about? Or would you say it's been pretty much continuous and undifferentiated?

Date: I think it's always been the same path. And I had no idea, of course, when I set out on that path how long it was going to be, but there's no reason to stop. I enjoy what I'm doing. I think I'm doing something useful. It's important, and it's fun. And I can't afford to retire, anyway.

Acceptance of the Relational Concept

Haigh: Now, at some point during the 1980s, you must have reached a situation where nobody's arguing with the big picture idea that in the future, everything will be relational and it's clearly superior. I know that you haven't run out of people to disagree with, but did that seem like a big shift, that now you weren't anymore arguing about relational versus non-relational?

Date: It might sound arrogant, but it didn't seem like a big shift because we always knew it would happen. It was only a question of when. It took a bit longer than we anticipated, but we knew. We knew it was the right way to go. If I had known how long it was going to take ahead of time, maybe I wouldn't have embarked on it. But no, we didn't pop the champagne corks or anything. You know, it was just another milestone which we knew was going to come along.

Haigh: And after you left the Codd and Date companies, did your relationship with Ted Codd continue very much as it had done before?

Date: Well, yes, except that he quite quickly went into a sort of semi-retirement. As he said himself, he wanted to spend more time with his grandchildren, and he really stopped producing original stuff and did not do any more consulting. He did a little and he worked on some things, but it was a very low level of activity then. And he moved to Florida, so instead of our being in the same state, California, he was now on the other side of the country so I didn't see him as much any more, either. We would run into each other at conferences occasionally. Things like that, you know. And we would talk on the phone sometimes.

Haigh: Okay. Let's return to your book output. We've talked about the SQL standard volumes.

Date: Yes. And there were a series of books on products. As I mentioned, the first thing I did when I left IBM was work on this book on DB2, *A Guide to DB2*, which was an obvious book to write. And that led to a few others on other products – SQL/DS and Ingres. The Ingres folks, in fact, approached me and asked if I would do a book for them under a consulting contract, which is what I did. In a way, it was a mistake because I got tarred by the brush of being part of Ingres and not being impartial; I was supposed to be independent. But I enjoyed doing it because I had a soft spot for the product. They did do some nice things. The optimization was good in the early days compared to the competition. But anyway, I wrote a series of product books. Then these other books, the ones I call the "Writings" books. Each one is basically a collection of various papers I have done in various different forums and pulled together and linked together in some cases. And I enjoy doing those because, in a sense, there is more of me in them. It's my stuff as opposed to just describing a product or even the relational model. So there's a bunch of those, and actually there's a new one coming out in a few weeks. I'm trying self-publishing on that one because I haven't had exactly a wonderful experience with the publishers of these books. They don't know how to sell them.

Haigh: You mean specifically the publishers of the "Writings" books?

Date: Yes. So, as I said, there's a new one coming out.

Haigh: You do seem to have been quite prolific.

Date: It's because I love writing. I get such a kick out of pushing the words around until they're right. And of course, two years later I look back and they're not right, so I have to do it again. But I just enjoy the process. I guess I'm very linear. I have to think things through on paper. That's they way I think, sorting ideas out on paper. And having done that exercise, if it's good, then let's preserve it.

Haigh: And then looking at some of your less formal writings and web pieces and so on from the last 10 years, it seems like you like arguing with people as well, right?

Date: No, I don't. I hate arguing, but people come at me with arguments and I have to defend myself and I have to defend the theory that is the basis of my livelihood and that I believe in. But I try to do it professionally, certainly, and politely wherever possible.

Haigh: And there's this debunking website that you're associated with?

Date: Yes. It's not my website, though. But it's funny: the guy whose website it is, Fabian Pascal, he doesn't agree with me about being polite, and so he calls a spade a spade. And if he thinks someone's a fool, he tells them so. And I don't agree with that. But because we're both doing stuff on that website – him more than me, but both of us – we get tarred with the same brush. And people confuse us, in fact, you know, accuse Fabian of writing things that I wrote or the other way around. And if I had known that that effect would occur, I'm not sure I would have done the pieces on that website, frankly, because I don't care to be seen as argumentative, because I don't think I am.

Haigh: Would you say "debate," rather than argue?

Date: But I don't even enjoy that. All I want to do is just try and explain to people and set the record straight. If somebody makes crazy remarks about third normal form or something and I say, "Well, actually this is what it is, it's like this." I'm just trying to explain it. I usually will not say, "Look, you're wrong," unless somebody is particularly persistent in their error.

Haigh: So In terms of the big picture, when you look back over the success that relational technology has had do you feel that in spite of niggles and blemishes here and there, relational technology has lived up to its promise? Even though established technologies are not perfect and there are still many people who misunderstand fundamental relational concepts, perhaps an inevitable consequence with relational technology so widely used. But

Date: No, absolutely not and I blame SQL for that. Partly I blame myself too for not seeing the problems and speaking out louder, earlier. But there are just so many things that are wrong with SQL. I don't want to get into a long discussion of that, because that could be a whole separate discussion. But it's true that few people use SQL anyway. I mean you use other front ends that map to SQL. SQL's become a target language instead of a source language. But it's bad as a target for all the same reasons it was bad as a source language. If I was writing some kind of front end I would prefer not to have to generate SQL. I would prefer to generate something clean, relational algebra or something. Can I talk about *The Third Manifesto* briefly?

Haigh: Yes, I think that that probably ties in with this.

The Third Manifesto

Date: *The Third Manifesto* is a very silly title, and I'll explain that title in a moment, but it is something that Hugh Darwen and I have been working on in a sense right from the beginning of our careers – certainly from about 1990 or so. The initial spur was that we saw this so-called object/relational thing coming along. And we saw some vendors, some people, making what we regarded as a huge technical mistake. And so we wrote the first draft of *The Third Manifesto* to explain what the mistake was and to try and get people to avoid it. And the first draft, the manifesto proper, was about 11 pages. But they were very terse, very hard to understand. So we wrote a whole book to explain it! The book was about 400 pages. And because the spur was this object/relational stuff, we originally called the book something like Foundation for Object/Relational Systems: *The Third Manifesto*.

Let me explain the "Third Manifesto" bit. There had been two previous manifestoes. There was one called *The Object-Oriented Database System Manifesto* in about 1989, and that proposed a set of prescriptions that a product would have to abide by in order to qualify for the label "Object Database System" according to the authors. Now I remember reading that one at the time and writing across the top "Nice Try But No Cigar." There are many detailed things I didn't like but the biggest thing was that they never mentioned the relational model. Now the relational model is the foundation of the database field conceptually. It was invented in 1969. This paper's published 20 years later. And a paper of this nature does not even mention it. So that was a huge problem. And it wasn't only me. Other people had problems with that one too. So, a bit later another manifesto came out. That was called *The Third Generation Database System Manifesto*.

Haigh: And who was the author of that?

Date: Well, both of them were by groups of people. The object-oriented one was a team led by Malcolm Atkinson, and the second one, which you can regard as a kind of rebuttal to the first, was a team led by Mike Stonebraker. And I should explain that second title. The idea was first generation database systems were hierarchies and networks. Second generation was SQL. Third generation is whatever comes next. And, they said categorically that whatever does come next must support the relational model. So that's good. Unfortunately they then went on to say that supporting the relational model meant supporting SQL. As Mike Stonebraker famously said, "SQL is intergalactic data-speak." Well of course we had a problem with that, Hugh and I. That is to say we felt that SQL is just not robust enough to provide the kind of firm foundation that we need for the foreseeable future. We really think we have to go back to the relational model. As I have indicated, there are huge differences between SQL and the relational model. So we wrote our own take on it and we called it *The Third Manifesto* because we wanted it to be judged alongside the other two. So the title was sort of chosen for us. The *Foundation for Object/Relational* name was a mistake. We called it that because object/relational was hot at the time. But in fact, object/relational so far has not really come to anything really major in the marketplace. So when we did a second edition, we called it *Foundation for Future Database Systems: The Third Manifesto*. That was a mistake too

because the problem with the future is it never arrives. So in the third one we tried to do two things. We tried to make the book more suitable as a textbook for serious students, and we tried to give it a title that more accurately captured what was in the book – *Databases, Types, and the Relational Model: The Third Manifesto*.

Haigh: What year was the first edition then?

Date: 1998 I think, and the second in 2000 and the third, 2006.

Haigh: And title aside, did people in the database management system research community read it alongside these other two manifestoes and give it the amount of attention you think it deserves?

Date: Well, I don't know if they read it alongside and in a sense now I don't really care very much because the other two have gone the way of all flesh. No one pays any attention to them. But on *The Third Manifesto* we do have little pockets of people all around the planet that are fans, if I may use that word. We have people who are building systems based on our ideas. There is at least one product based on the ideas. People are doing further research and using it as a teaching vehicle and so on. So we have little groups of fans even among some of the vendors, I can't say all but certainly at some of the vendors, there are little groups of people who are saying, "What we should really be doing is this Third Manifesto stuff."

We are proposing in *The Third Manifesto*, first of all, that somebody build a real relational system as opposed to an SQL system. That it should support the relational model, all of it, not just bits. And it should have a decent relational language, not SQL. Now we are not so stupid as to think that SQL is ever going to go away. So we have to worry about what I like to call the SQL Legacy (being optimistic). But we draw a parallel with Cobol. In the application development world Cobol has not gone away and never will. But we don't use Cobol to develop applications if we can avoid it. There are much better ways to build applications than using Cobol. So we see SQL as a kind of database Cobol. It's not going to go away, but if we can produce an interface that truly is significantly better then people will migrate to it. And by a process of attrition, the use of SQL will atrophy somewhat. It won't go away. It will always be there. But if there's something better alongside it then good things will happen. And part of making that happen is we have to have a migration route from SQL to whatever this new thing is. We thought about that very hard. And in the book, *The Third Manifesto* book, we describe how we see the migration working. And we recognize that the effort will probably fail, but we're optimists. We take the view that if we do nothing then nothing will happen, but if we do something then something might happen. And we're pushing very hard. In the U.K. and Europe, Hugh has presented these ideas at many universities. I have presented them to vendors and universities over here, and as I say, we do have little groups of people who agree

that this would be a good thing to do. Of course, the numbers are tiny. But it's so much more intellectually satisfying than having to deal with some of what's out there today.

Haigh: You seem to have been particularly productive in the last few years.

Date: Well that's true. But one of the things about *The Third Manifesto* is that we truly see it as what should be the foundation. And so of course we're developing things ourselves on top. One of the big things we have is a theory of types. The relational model requires types but it doesn't say anything about them. So there's a whole orthogonal theory that's needed there. And we developed that. That theory includes a model of type inheritance which is an interesting subject in its own right. More recently we've taken the ideas of the manifesto and applied them to the problem of time, the time dimension. How do you handle time in databases? Turns out it's an amazingly complex problem. And I'd always sort of vaguely assumed that we'd have to do something new and different to make time work. And to my great joy we discovered that you can do it all in the relational model. The relational model has the basics of what you need. You need some good shorthands, but they are ultimately only shorthands for things you can already do in the relational model. So there's a good basis there and we wrote a book about that one. We have some other ideas that we're working on, actual true extensions to the relational model – very preliminary ideas there – that all build on the work in *The Third Manifesto*. So we find it a tremendous platform. And yes, I should say, in my seminars it's all that I teach. I don't use SQL for examples. I use the language we have in *The Third Manifesto*, and people never have difficulty understanding it. And it is so much cleaner and properly closed and all those good things. So certainly, I think all of these books from that time on use the ideas of the Manifesto, except possibly one – no, even that one does as well. I don't want to talk about all these books in depth, but this was a fun one, the dictionary. Have you seen that book?

Haigh: No.

Date: It's this little pocket book. And it wasn't my idea. A friend of mine, Jonathan Gennick, said he'd been thinking about a book. Well, let me back off. At home I have a huge library of my own stuff, math books, computer books, database books and so on. But there's one book I pull off the shelf almost every day, and it's a dictionary of mathematics. And it's not because I don't know something, it's because I want a good, crisp definition. There's a big difference between knowing something and being able to define it precisely! And it occurred to me, through the help of this friend, that there was nothing like that in the database world. So, I decided to write one. I had great fun doing that and I wanted it to be literally a pocket book and cheap. People could buy it as an impulse buy and they could keep it in their pocket. I think it's done okay. The trouble, is dictionaries grow all the time and I really want to do a second edition. Well, I have done it, but I haven't committed it to the publisher yet, and it's expanded by about 33 percent.

Haigh: You need a larger pocket.

Date: But it'll still be under 200 pages. It'll still be small – a pocket book. And let me just mention this last book. There's a new implementation technology on the horizon, something that is radically different from what all these guys are doing today in IBM, Oracle and so on, totally different. And the company that the technology belongs to hired me about five or six years ago to help them get the word out. Now when they approached me I said, "I'm independent. I have got to be careful here." They said, "No really, we'd like you to see what we have." So they came out. They were in New York at the time and they came out to California. I met with them. They gave me the demo, gave me presentations. We had discussions for a day or so. But of course when you see demos and things you never really know what you're looking at. So I said, "Well it looks interesting but I don't know. Can I have a copy of the patent and I'll read that and I'll get back to you in a month or so?" Okay, they gave me the patent. I don't know if you've ever read patents. They're written to be hard to understand. God, it was awful. But I read it several times. Each time through I got a little bit more and I began to realize that this is terrific stuff. This is really great. So eventually I called them back and said, "Yes, I think I can help you because I think the technology is wonderful. I'd love to be able to go and talk about it at conferences and so on. And one thing I can do is write a book for you." They said, "Great." So I wrote the book. The book exists. It has been written. And then what happens? The company fell apart. Some people got very greedy. Infighting developed and I'll spare you all the horrible details. But because of that, the book, the manuscript, is still sitting on the shelf waiting for the company to sort itself out.

Haigh: Can you say what the company's called?

Date: It's called Required Technologies. And their technology is called the TransRelational Model. And I have to make it very clear. It's a model. It's abstract. It can be realized physically in many different ways, but it's at a lower level than the relational model. The idea is you'd map the constructs of the relational model into transrelational constructs and then down to whatever is underneath. Trans here doesn't mean beyond, it means transform. Transform relational concepts into these concepts.

Haigh: **"**Sub-relational" didn't sound as good!

Date: Well, you can imagine the arguments we went through trying to come up with a name for this. And I wasn't terribly happy about transrelational because it sounds like it's superseding relational. But anyway, I don't know what's going to happen to the company. It's possible the company will totally die. But the technology is so great, it's going to survive somehow. And it has all kinds of wonderful characteristics. Quite apart from anything else it's blindingly fast. I mean orders of magnitude faster than the current systems. And it's scalable. So to join 20 relations takes only twice the time it takes to join 10. And if it had no other

advantages, that one right there is huge, because normally we're talking about multiplicative costs. But these are additive. It's absolutely brilliant in my opinion. I can't talk much about it because I'm still under a non-disclosure agreement. But they do have what they call the initial patent, and by definition that's in the public domain. And I can talk about that, and I do in my seminars. And, oh I should tell you: Ted Codd saw this technology before he died. And he looked at it and said, "That's what I meant." He was very enthusiastic about it. In fact, he wrote a letter to the company saying that, "This was the sort of thing I had in mind all along." So, it's so frustrating that it isn't out there yet and, for me, that the book isn't out there yet.

Haigh: Is there anything else that you want to say about your recent work, current projects, or plans for the future?

Date: Plans for the future are more of the same. I mean new problems keep coming along. I'd like to look at how the ideas that we have in our temporal data work would extend to spatial data. You know there has to be a lot of commonality there. So that's on the agenda to look at. There's this idea I mentioned of possible extensions to the relational model. I'm being very tentative about that because the relational model is such a fine thing. But there are some things it doesn't do. And we have some preliminary ideas. The relational model is a piece of science. Although Ted Codd invented it, it doesn't belong to him, it belongs to science. Anybody who's capable can make contributions to it. So we are operating in that sort of vein; anybody who can come up with some new ideas and so on that fit with it, they are welcome to do so. It's like mathematics. Anyone can extend mathematics. It is a piece of mathematics – that's exactly what it is.

Haigh: Then I will finish with two open-ended questions. Looking back over the whole of your career to date, what would you say is your biggest regret either in terms of something that you personally did or didn't do or just in terms of how something turned out in the world?

Date: Well, of course, the regret has to be that the true promise of the relational model has not yet been fully realized. And that is really due to a combination of factors. The SQL language is a very imperfect concrete realization of the abstract ideas and I wish I had realized how bad it was sooner. So I blame myself partly for that. It's ironic. As Ken Jacobs said this morning, it is true that I went to what became the SQL committee, the ANSI SQL committee, at the invitation of the IBM rep to tell them what IBM SQL was about. So in some ways it's all my fault. So of course I regret that. While I'm pleased that these SQL systems are now ubiquitous, I wish they were relational systems. So that has to be the big regret.

Haigh: What's the accomplishment for which you feel most satisfaction?

Date: And that's an easy one too, isn't it? Of course I'm pleased that I've had the influence that I have through my writings and teachings. And just this week people have been

coming up and saying, “Well, Ted was a genius but we didn’t understand his paper, but of course we had your book.” And I say, “Thank you, that’s very nice.” But I’m most proud of the fact that I’ve been able to communicate this stuff to a lot of people and therefore help spread the technology. I only wish it was done better!

Haigh: OK, so that would conclude the interview unless you have anything else to say.

Date: I don’t think so. Thank you very much for conducting the interview.

Haigh: And thank you very much for taking part.

Date: I was pleased to.

C. J. DATE: BIOGRAPHICAL NOTES

Chris Date is an independent author, lecturer, researcher, and. consultant, specializing in relational database technology (a field he helped pioneer). He is best known for his books, especially *An Introduction to Database Systems* (see below), and for his database lecture and seminar series. He resides in Healdsburg, California.

Mr. Date was born in England in 1941. He began his professional career in 1962 as a mathematical programmer with LEO Computers, London, England (later part of ICL, now Fujitsu). In 1967 he moved to the IBM Development Laboratory, Hursley, England (at that time responsible for the PL/I mission within IBM), where he was involved in advanced technology and design work on the integration of database functionality into PL/I. He was also instrumental at Hursley in establishing the IBM European Laboratories Intermediate Professional Training program (ELIPT); as part of that program, he developed courses in systems programming techniques, performance measurement, operating system externals and internals, and database concepts, and taught those courses at IBM locations throughout Western Europe.

In 1974 Mr. Date moved to California, first to the IBM Programming Center in Palo Alto and subsequently to the IBM Santa Teresa Laboratory in San Jose when that laboratory opened in 1977. During that period he was responsible for the design of a database language known as UDL (Unified Database Language), which provided support for all three of the then well known database approaches (relational, hierarchic, network) in a uniform and consistent manner. That activity involved the development of a series of prototype implementations of various subsets of the language, together with a continuing dialog with user groups in GUIDE, SHARE, and elsewhere to ensure that the design of the language reflected genuine user needs. Subsequently, Mr. Date worked on technical planning and externals design for the influential database products SQL/DS (announced in 1981 for the VSE environment and 1983 for VM) and DB2 (announced in1983 for MVS). He left IBM in May 1983.

Mr. Date has been active in the field of database for nearly 40 years He was one of the first people anywhere to recognize the significance of Codd's pioneering work on the relational model, and has done more than anyone else to make that work accessible to others. His book *An introduction to Database Systems* (first edition 1975, eighth edition 2004) is the standard text on the subject; it has sold over three quarters of a million copies (not including translations) and is used by several hundred colleges and universities worldwide. In 1982 he published a sequel to that book (Volume II), covering a range of more advanced aspects of database technology. His complete list of books is as follows:

- *An introduction to Database Systems*: *Volume I* (1975; 2nd edition 1977, 3rd edition 1981, 4th edition 1985, 5th edition 1990, 6th edition 1995, 7th edition 2000, 8th edition 2004)

- *An introduction to Database Systems*: *Volume II* (l983)
- *Database: A Primer* (1983)
- *A Guide to DB2* (1984; 2nd edition 1988, 3rd edition 1989, 4th edition 1992; Colin J. White, coauthor, for 2nd and subsequent editions)
- *A Guide to INGRES* (1987)
- *A Guide to the SQL Standard* (1987; 2nd edition 1989, 3rd edition 1993, 4th edition 1996; Hugh Darwen, coauthor, for 3rd and 4th editions)
- *A Guide to SQL/DS* (1988; Colin J. White, coauthor)
- *A Guide to SYBASE and SQL Server* (1992; David McGoveran, coauthor)
- *Relational Database: Selected Writings* (1986)
- *Relational Database Writings 1985-1989* (1990)
- *Relational Database Writings 1989-1991* (1992; Hugh Darwen, coauthor)
- *Relational Database Writings 1991-1994* (1995)
- *Relational Database Writings 1994-1997* (1998: Hugh Darwen and David McGoveran, coauthors)
- *Relational Database Writings 1997-2000* (2006)
- *Relational Database Writings 2000-2006* (2006)
- *Databases, Types, and the Relational Model: The Third Manifesto* (1998; 2nd edition 2000; 3rd edition 2006; Hugh Darwen, coauthor)
- *WHAT Not HOW: The Business Rules Approach to Application Development* (2000)
- *The Database Relational Model: A Retrospective Review and Analysis* (2001)
- *Temporal Data and the Relational Model* (2003; Hugh Darwen and Nikos A. Lorentzos, coauthors)
- *Database in Depth: Relational Theory for Practitioners* (2005)
- *The Relational Database Dictionary* (2006; 2nd edition to appear)
- *Logic and Databases: The Roots of Relational Theory* (2007)
- *Go Faster! The TransRelational [tm] Approach to DBMS Implementation* (to appear)

His books have been translated into many languages, including Chinese, Dutch, French, German, Greek, Italian, Japanese, Korean, Polish, Portuguese, Romanian, Russian, Spanish, and Braille. Mr. Date has also produced well over 400 research papers and technical articles and has made a variety of original contributions to database theory (in the fields of database design, view updating, and integrity constraints among others). For the past several years he has been working with a close colleague, Hugh Darwen, on *The Third Manifesto* (a detailed proposal for the future of data and database management systems); on an abstract and robust model for type inheritance; and (more recently) on temporal database support.

Mr. Date has lectured widely on technical subjects--principally on database topics, and especially on relational database --throughout the United States and also in Europe, Australia, Latin America, and the Far East. He enjoys a reputation that is second to none for his ability to communicate the complexities of database technology in a clear and simple fashion, both on paper and in live presentations. As a result, he is in great demand as a lecturer and keynote presenter at technical conferences, professional development seminars, user group meetings, and the like.

Mr. Date was a cofounder of the well known consulting company Codd and Date International and various subsidiary companies. He has provided technical consulting services to numerous clients, especially database system vendors, including ADR, Computer Associates, Cincom, DEC, IBM, ICL/Fujitsu, Ingres, NCR, Oracle, Required Technologies, Sybase, Tandem, Teradata, and Versata.

Mr. Date holds an Honours Degree in Mathematics from Cambridge University, England (BA 1962, MA 1966). He also holds the honorary degree of Doctor of Technology from De Montfort University, England (1994). He is a member of ACM, the ACM Special Interest Group on Management of Data (SIGMOD), and the British Computer Society (BCS). He was made an Honorary Fellow of The Irish Computer Society in 2003 and was inducted into the Computing Industry Hall of Fame in 2004. He was a regular columnist for the magazines *Database Programming & Design* and *Intelligent Enterprise* for some ten years (1992-2001); since 2000, he has also been an occasional columnist for *DataToKnowledge Commentary*. For several years he was also a regular contributor to the website *www.dbdebunk.com*.