Edgar F. Codd

August 23rd, 1923 - April 18th, 2003

a tribute and personal memoir

C. J. Date

This is an expanded version of a piece that originally appeared (and is still available) on the SIGMOD website *www.acm.org/sigmod*. Here's the opening paragraph from that earlier piece:

> By now there cannot be many in the database community who are unaware that, sadly, Dr. E. F. Codd passed away on April 18th, 2003. He was 79. Dr. Codd, known universally to his colleagues and friends—among whom I was proud to count myself—as Ted, was the man who, singlehanded, put the field of database management on a solid scientific footing. The entire relational database industry, now worth many billions of dollars a year, owes the fact of its existence to Ted's original work, and the same is true of all of the huge number of relational database research and teaching programs under way worldwide in universities and similar organizations. Indeed, all of us who work in this field owe our career and livelihood to the giant contributions Ted made during the period from the late 1960s to the early 1980s. We all owe him a huge debt. This tribute to Ted and his achievements is offered in recognition of that debt.

However, I've discovered I was operating on a false assumption when I wrote this paragraph. To be more specific, I've found—and it's a sad comment on the state of our field that I feel I have to say this—that many database practitioners, and even some researchers, really don't know who Ted was or what he did. In my own day-to-day database activities (for example, on the seminars I teach), I often encounter people who've never even heard of him! So I thought it would be a good idea to amplify and republish my original tribute: in particular, to elaborate briefly on the nature of some of Ted's numerous technical contributions. I'd also like to say a little more about "Ted the man"—i.e., Ted Codd as I personally remember him, and what he meant to me.

BACKGROUND

Ted Codd was a native of England and a Royal Air Force veteran of World War II. He moved to the United States after the war and became a naturalized US citizen. He held MA degrees in mathematics and chemistry from Oxford University and MS and PhD degrees in communication sciences from the University of Michigan.

Ted began his computing career in 1949 as a programming mathematician for IBM on the Selective Sequence Electronic Calculator. He subsequently participated in the development of several important IBM products, including the 701, IBM's first commercial electronic computer, and STRETCH, which led to IBM's 7090 mainframe technology. Then, in the late 1960s, he turned his attention to the problem of database management—and over the next several years he

created the invention with which his name will forever be associated: **the relational model of data**.

DATABASE CONTRIBUTIONS

Ted Codd's relational model is widely recognized as one of the great technical innovations of the 20th century. Ted described it and explored its implications in a series of research papers ── staggering in their originality──that he published during the period from 1969 to 1981. The effect of those papers was twofold:

- First, they changed for good the way the IT world perceived the database management problem.

- Second, as already mentioned, they laid the foundation for a whole new industry.

In fact, they provided the basis for a technology that has had, and continues to have, a major impact on the very fabric of our society. It's no exaggeration to say that Ted is the intellectual father of the modern database field.

To give some idea of the scope and extent of Ted's accomplishments, I'd like to highlight in this section what seem to me the most significant of his database contributions. Of course, the biggest of all was, as already mentioned, to make database management into a science (and thereby to introduce a welcome and sorely needed note of clarity and rigor into the field): The relational model provides a theoretical framework within which a variety of important problems can be attacked in a scientific manner. Ted first described his model in 1969 in an IBM Research Report:

- "Derivability, Redundancy, and Consistency of Relations Stored in Large Data Banks," IBM Research Report RJ599 (August 19th, 1969)

He also published a revised version of this paper the following year:

- "A Relational Model of Data for Large Shared Data Banks," *CACM 13,* No. 6 (June 1970) and elsewhere[1].

(This latter──i.e., the 1970 paper──is usually credited with being the seminal paper in the field, though this characterization is a little unfair to its 1969 predecessor.) Almost all of the novel ideas described in outline in the remainder of this section, as well as numerous subsequent technical developments, were foreshadowed or at least hinted at in these first two papers; in fact, some of those ideas remain less than fully explored to this day. In my opinion, everyone professionally involved in database management should read, and reread, at least one of these papers every year. (It's true they're not all that *easy* to read, being fairly abstract and somewhat mathematical in tone. On the other hand, they stand up extremely well to being read, and indeed

[1]Many of Ted's papers were published in several places. Here I'll just give the primary sources.

repeatedly reread, over 30 years after they were written! How many technical papers can you say that of?)

Incidentally, it's not as widely known as it should be that Ted not only invented the relational model in particular, he invented the whole concept of a *data model* in general. See his paper:

- "Data Models in Database Management," *ACM SIGMOD Record 11*, No. 2 (February 1981)

This was the first of Ted's papers to include a definition of the term *data model*. It also addressed the question: What purposes are data models in general, and the relational model in particular, intended to serve? And it then went on to offer evidence in support of the claim that, contrary to popular belief, the relational model was the first data model to be defined. (The so-called hierarchic and network models, which are often thought to have predated the relational model, were actually defined after the fact by a process of induction──in this context, a polite word for guesswork──from existing implementations.)

In connection with both the relational model in particular and data models in general, the paper also stressed the importance of the distinction, regrettably underappreciated even today, between a data model and its physical implementation.

Ted also saw the potential of using *predicate logic* as a foundation for a database language. He discussed this possibility briefly in his 1969 and 1970 papers, and then, using the predicate logic idea as a basis, went on to describe in detail what was probably the very first relational language to be defined, *Data Sublanguage ALPHA,* in:

- "A Data Base Sublanguage Founded on the Relational Calculus," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control*, San Diego, Calif. (November 1971)

ALPHA as such was never implemented, but it was extremely influential on certain other languages that were, including in particular the Ingres language QUEL and (to a much lesser extent) the IBM language SQL as well.

If I might be permitted a personal anecdote at this point, I remember being invited, in 1974, to Anaheim, California, to present the basic ideas of the relational model to the GUIDE Database Language Working Group. (I probably don't have that title quite right, but it was a project within GUIDE, the IBM user group, to examine possible programming language extensions for database access.) Members of the working group had coded five sample applications using various database products of the time (TOTAL, IDS, IMS, etc.), and the meeting began with those various solutions being presented. I distinctly remember the group's reaction (chiefly amazement, mingled with delight) when I was able to show that each of their applications──which I came to stone cold, never having seen them before that afternoon──involved *just one line* of ALPHA code! At that point, the IBM representative to the group took me off to one side and told me "not to come on too strong about this relational stuff."

I said: "Why not?" He said: "Because we don't have a product." To which I answered: "Then maybe we should get one." SQL/DS was announced a mere seven years later ...

Back to Ted. Ted subsequently defined the *relational calculus* more formally, as well as the *relational algebra,* in:

- "Relational Completeness of Data Base Sublanguages," in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6* (Prentice-Hall, 1972)

Very loosely speaking, relational calculus provides a notation for *defining* some desired relation (typically the result of some query) in terms of others, while relational algebra provides a set of operators for *computing* some desired relation from others. Clearly, each could be used as a basis on which to define a query language. As I've already indicated, QUEL is an example of a language that's based on the calculus. SQL, by contrast, is a mixture: It includes some elements that are calculus-based, others that are algebra-based, and still others that are neither.

As the title indicates, the paper also introduced the notion of *relational completeness* as a basic measure of the expressive power of a database language. Essentially, a language is said to be relationally complete if it's as powerful as the calculus. To quote: "A query language ... which is claimed to be general purpose should be at least relationally complete in the sense defined in this paper. [Such a language need never resort] to programming loops or any other form of branched execution—an important consideration when interrogating a [database] from a terminal."

The same paper also described an algorithm—*Codd's reduction algorithm*—for transforming an arbitrary expression of the calculus into an equivalent expression in the algebra, thereby (a) proving the algebra is relationally complete and (b) providing a basis for implementing the calculus.

Ted also introduced the concept of *functional dependence* and defined the first three *normal forms* (1NF, 2NF, 3NF). See the papers:

- "Normalized Data Base Structure: A Brief Tutorial," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access, and Control*, San Diego, Calif. (November 11th-12th, 1971)

- "Further Normalization of the Data Base Relational Model," in Randall J. Rustin (ed.), *Data Base Systems: Courant Computer Science Symposia Series 6* (Prentice-Hall, 1972)

Simplifying considerably, we can say that:

- Given relation *R,* attribute *B* of *R* is functionally dependent on attribute *A* of *R*—equivalently, the functional dependence (FD) $A \rightarrow B$ holds in *R*—if and only if, whenever two tuples of *R* have the same value for *A,* they also have the same value for *B*.

- Relation *R* is in third normal form, 3NF, if and only if the only FDs that hold in *R* are of the form $K \rightarrow B$, where *K* is a key. (In the interests of accuracy, I should say that this is really a definition of what subsequently became known as *Boyce/Codd normal form*, BCNF, not 3NF as such. Part of the confusion arises from the fact that Ted himself referred to BCNF as "third" normal form for some years.)

Between them, Ted's two normalization papers laid the foundations for the entire field of what is now known as *dependency theory*, an important branch of database science in its own right. Among other things, that theory serves as a basis for a truly scientific approach to the problem of logical database design.

As an aside, I note that the first of the two papers, the tutorial one, also includes arguments for excluding pointers from the user's view of the database. The relational model's total ban on pointers is just one of many factors that sharply distinguish it from its competitors. Indeed, any language that exposes pointers to the user, in any shape or form, forfeits all right to be called relational—and the introduction of pointers into the SQL:1999 standard in particular can thus only be deplored. But I digress.

There's something else I'd like to say here. Normalization is sometimes criticized on the grounds that "it's all really just common sense"; any competent professional would "naturally" design databases to be fully normalized anyway, without having to know anything about dependency theory at all. In other words, the problems with less than fully normalized designs are "obvious," and common sense is sufficient to tell us that fully normalized designs are better. But what do we mean by "common sense"? What are the *principles* (inside the designer's brain) that the designer is applying when he or she chooses a fully normalized design? The answer is, of course, that they're exactly the principles of normalization. In other words, those principles are indeed just common sense—but (and here comes the point) they're *formalized* common sense. The whole point is to try to identify such principles and formalize them—which isn't an easy thing to do! But if it can be done, then we can *mechanize* them; in other words, we can get the machine to do the work. Critics of normalization usually miss this point; they claim, quite rightly, that the ideas are all basically common sense, but they typically don't realize that it's a significant achievement to state what "common sense" means in a precise and formal way.

Yet another crucial notion introduced by Ted was that of *essentiality*. Ted defined that notion in:

- "Interactive Support for Nonprogrammers: The Relational and Network Approaches," *Proc. ACM SIGMOD Workshop on Data Description, Access, and Control, Vol. II*, Ann Arbor, Michigan (May 1974)

This paper was Ted's principal written contribution to "The Great Debate." The Great Debate—the official title was *Data Models: Data-Structure-Set vs. Relational*—was a special event held at the 1974 SIGMOD Workshop; it was subsequently characterized in *CACM* by Robert L. Ashenhurst as "a milestone event of the kind too seldom witnessed in our field." (I should explain that "the data-structure-set model" is just the CODASYL network model by another name.)

The concept of essentiality, introduced by Ted in this debate, is a great aid to clear thinking in discussions regarding the nature of data and DBMSs. Basically, an "information carrier" (in other words, a construct for the representation of data) is said to be *essential* if and only if its removal would cause a loss of information. Now, the relational model provides just one way to represent data—namely, by means of relations themselves—and the sole essential information carriers in a relational database are thus necessarily relations, *a fortiori*. By contrast, other data models typically provide many distinct ways to represent data (lists, bags, links, sets, arrays, and so on), and all or any of those ways can be used as essential information carriers in a nonrelational database. One way of representing data is both necessary and sufficient; more than one introduces complexity, but no additional power.

As a matter of fact, it's the concept of essentiality that forms the underpinning for the well-known, and important, *Information Principle:*

> *The entire information content of a relational database is represented in one and only one way: namely, as attribute values within tuples within relations.*

I heard Ted refer to this principle on more than one occasion as *the* fundamental principle underlying the relational model. (I remark in passing that a better name for it might be *The Principle of Uniform Representation*.)

It seems appropriate to round out this discussion with a succinct, reasonably formal definition of the relational model. (This definition doesn't come from any of Ted's papers—it's my own attempt to capture the essence of what Ted was talking about in all of the papers I've been discussing, as well as many others.) Briefly, the relational model consists of five components:

1. An open-ended collection of **scalar types** (including in particular the type *boolean* or *truth value*)

2. A **relation type generator** and an intended interpretation for relations of types generated thereby

3. Facilities for defining **relation variables** of such generated relation types

4. A **relational assignment** operation for assigning relation values to such relation variables

5. An open-ended collection of generic **relational operators** ("the relational algebra") for deriving relation values from other relation values

Let me conclude this section by observing that Ted's work on the relational model didn't just set the entire field of database management on a solid scientific footing—it also provided the foundations for the careers of numerous people, myself not least (and doubtless many readers of this tribute could say the same). Not to mention the fact that there is, and continues to be, tremendous intellectual excitement, rigor, and integrity, in those foundations that Ted laid down.

(By the way, I don't want to give the impression that all of the foundation-level problems have been solved. They haven't. That's partly why the field is still so rewarding!) So those of us who have continued in Ted's footsteps haven't just had a career and a livelihood, as I said before—we've had work that was, and continues to be, both interesting and useful. We all owe Ted an enormous debt of gratitude.

OTHER CONTRIBUTIONS

Ted Codd's achievements with the relational model shouldn't be allowed to eclipse the fact that he made major research contributions in several other important areas as well, including *cellular automata, multiprogramming,* and *natural language processing*, to name just three. Other people are better qualified than I to comment on those contributions; all I'd like to do here is draw attention to some of the key references. First, the work on cellular automata is documented in a monograph by Ted:

- *Cellular Automata,* Academic Press (1968)

Second, Ted in fact led the team that developed IBM's very first multiprogramming system. He reported on that work in:

- "Multiprogramming STRETCH: Feasibility Considerations" (with three coauthors), *CACM 2,* No. 11 (November 1959)

- "Multiprogram Scheduling," Parts 1 and 2, *CACM 3,* No. 6 (June 1960); Parts 3 and 4, *CACM 3,* No. 7 (July 1960)

As for his work on natural language processing, see among other writings the paper:

- "Seven Steps to Rendezvous with the Casual User," in J. W. Klimbie and K. L. Koffeman (eds.), *Data Base Management,* Proc. IFIP TC-2 Working Conference on Data Base Management (North-Holland, 1974)

Other research areas to which Ted contributed include *online analytical processing* (OLAP) and *business automation*.

In addition to all of his research activities, Ted was professionally active in several other areas as well. In particular, it doesn't seem to be widely known that he was the founder of SIGMOD! (More precisely, he founded the ACM Special Interest Committee on File Description and Translation, SICFIDET, which later became an ACM Special Interest Group, SIGFIDET, and subsequently changed its name to the Special Interest Group on Management of Data, SIGMOD.) He was also tireless in his efforts, both inside and outside IBM, to obtain the level of acceptance for the relational model that he rightly believed it deserved: efforts that were, of course, eventually crowned with success.

Note: I think perhaps I should qualify the foregoing remarks a little. It's true that the relational model is now widely accepted. At the same time ... I don't want to appear churlish,

but I feel bound to say how much I regret the fact (as I'm quite sure Ted did too) that the model still hasn't been implemented either faithfully or properly. To elaborate very briefly:

- *Faithful implementation:* Today's products don't implement the model faithfully. Instead, they suffer from sins of both omission and commission: Certain important relational features aren't supported at all, while other nonrelational features are.

- *Proper implementation:* Today's products also fail in all too many ways to deliver on the full promise of the relational model (regarding data independence, for example).

Further discussion of such matters would clearly be inappropriate here, but I think it would be just as inappropriate not to mention them at all.

PERSONAL MEMORIES

I first met Ted in 1971. I'd been doing some database work myself for IBM in England in 1970-71, as a result of which I entered into a correspondence with Ted, and he invited me to the US to present my ideas at various locations within IBM. So I got to meet Ted at the IBM San Jose Research Laboratory, where he was working at the time. I was immediately struck by his energy, and in particular by the care and precision with which he expressed himself. He was always careful never to overclaim. For example, in his paper "Further Normalization of the Data Base Relational Model," he states that one advantage of normalization is that it "[tends] to capture some aspects of the semantics (minor, of course)." I love that parenthetical remark! What a contrast to some of the overblown claims we so often encounter elsewhere in the literature.

Following our meeting in 1971, Ted and I began a cooperation—at first informal, later more formal—that lasted essentially throughout the remainder of his career. The precise nature of that cooperation varied over time, but the essence of it was that Ted continued to do research on his model and I took it on myself to present and explain his ideas at technical conferences, user group meetings, and the like. In this connection, Ted was always encouraging, supportive, and generous to me, as indeed he was to all of his friends and coworkers. For example (this is from a letter to me dated April 7th, 1972): "I am pleased to learn you are contemplating writing a [database book] ... The kind of book you propose is much needed ... I feel confident that you can produce an excellent book. You may make use of any of the material in my papers for this purpose."

Other people who had the opportunity to work with or for Ted over the years have confirmed to me what an inspiration he could be in their own endeavors.

Ted was easily one of the most unforgettable characters I ever met. He was a genius, of course—and like all geniuses he "stood at a slight angle to the rest of the universe." It wasn't just database, either. Something else he displayed his genius in was *losing things* ... When we were working together in Codd and Date Inc. (a consulting company we and Sharon Weinberg, later Sharon Codd, started in the early 1980s), I once personally saw him lose a crucial document as he crossed from one side of our office to the other. I didn't see it go—I just saw him start off

with it on one side of the room and arrive at the other side without it. I don't think we ever found it again.

Another way his genius manifested itself was in a somewhat idiosyncratic sense of humor (again I suspect this is a property that is common to many geniuses). Things that most people didn't think were funny at all he would sometimes find incredibly funny; conversely, he would fail, sometimes, to see the funny side of things that other people thought were hilarious. When Sharon, Ted, and I were forming our consulting company, the question of a company logo came up, for use on business cards and the like. I said I thought the logo should incorporate a stylized picture of a table. Sharon agreed. Ted said: "I don't get it—we're not in the furniture business, are we?"

There are many, many Ted Codd stories, but I think one more here has to suffice. (I like this one, though actually I wasn't there when it happened. So it might be apocryphal. But it does have the ring of truth about it.) Apparently, when Ted was made an IBM Fellow—see the next section—he said:

> "It's the first time I recall someone being made an IBM Fellow for someone else's product."

CONCLUDING REMARKS

Ted was a genuine computing pioneer. And he achieved more in his lifetime than most of us ever will or could, if we had ten lifetimes. Look at the honors he was awarded (and this is only a partial list):

- IBM Fellow
- ACM Fellow
- Fellow of the British Computer Society
- Member of the National Academy of Engineering
- Member of the American Academy of Arts and Sciences
- 1981 ACM Turing Award
- Outstanding recognition award from IEEE
- First annual achievement award from IDUG
- 2001 Hall of Fame Lifetime Achievement Award from DAMA

All thoroughly deserved, of course.

Ted was an inspiration and a good friend to all of us who had the privilege, fortune, and honor to know him and work with him. It's a particular pleasure to be able to say that he was always scrupulous in giving credit to other people's contributions. Moreover, as I've already mentioned (and despite his huge achievements), he was also careful never to overclaim; he would never claim, for example, that the relational model could solve all possible problems or that it would last forever. And yet those who truly understand that model do believe that the class of problems it can solve is extraordinarily large and that it will endure for a very long time. Systems will still be being built on the basis of Codd's relational model for as far out as anyone can see.

Ted is survived by his wife Sharon; a daughter, Katherine; three sons, Ronald, Frank, and David; his first wife Libby, mother of his children; and six grandchildren. He also leaves other family members, friends, and colleagues all around the world. He is mourned and sorely missed by all.

—C. J. Date
Healdsburg, California, 2003

Chris Date has been working in the database field ever since 1970. He knew and collaborated with Ted Codd throughout the period from 1971 onward. He is the author of several database textbooks, including in particular The Database Relational Model: A Retrospective Review and Analysis *(Addison-Wesley, 2001), which is a historical account and assessment of Codd's development of the relational model. Portions of the present tribute are based on this book.*