The World as Database: On the Relation of Software Development, Query Methods, and Interpretative Independence

David Gugerli

In the early 1970s, conventional database management systems were challenged by relational database models. Relational search procedures, as proposed by Edgar F. Codd, offered both improved data independence and an enhanced combinatorial freedom for the user. At the same time, late modern concepts of an open text boosted a reader's interpretative independence, bringing to an end the heyday of hermeneutics. This politically significant change of technocultural patterns led to wild debates, and not only among intellectuals. Likewise, the change meant big trouble for software engineers. Advocates of user independence tried to promote their concepts, destabilize adversative suggestions, and gain allies by aggressive claims, heroic narratives, and cooperative publication strategies.

When the investigators in the American crime series *CSI* roll up their sleeves on televisions all over the world, they never experience a shortage of corpses. Indeed, most often, murders occur in seemingly unrelated multiples, calling on all the skills and equipment that these high-tech forensic experts can muster. Since September 2000, more than 260 episodes of the original *CSI* have been produced. Together with its companion programs, *CSI: Miami* and *CSI: New York*, the series has cornered the global market in television crime shows.[1] The settings, editing sequences, and dramatic pace of the show have long set the standard for the entertainment industry.

The success of *CSI* and similar programs derives from the techno-scientific portrayal of top-notch detective work that follows a strict formula. Steven Bochco, one of the most successful producers of American TV series, calls *CSI* a "formula show" that made an unpredictable break with previous recipes for success: "The characters have no private life, no history, and are in some sense indistinguishable from one another. Yet *CSI* is the most successful series in the world. In the last six years, *CSI* has changed the rules of the game. Before, productions were judged on whether they had subtexts and strong characters.

Information & Culture, Vol. 47, No. 3, 2012

DOI: 10.7560/IC47302

CSI showed that the opposite works better. Form instead of content."[2] The cultural impact of *CSI* on two hundred countries and hundreds of millions of enthusiastic viewers is difficult to assess and has become the controversial subject of a slew of studies. It is not quite clear how seriously defenders and jurors on the viewers' side of the screen take *CSI* proceedings. Criminology courses, on the other hand, are booming, and there may well be crimes that are planned and carried out in such a way as to resist detection even by methods as refined as *CSI*'s. This is all part of what is known as the *CSI* effect, or *CSI* syndrome.[3]

Discussion of the syndrome usually overlooks the fact that not only the settings of technoscientific TV crime shows but also the real forensic and criminal work have changed.[4] The profound shift in interpretative patterns of crime shows is, rather, a telling symptom of a general shift in dominant patterns and procedures for interpreting the world—on both sides of the TV screen.[5] I would even claim that the schemes followed by *CSI* are significant indicators for the dominant interpretative culture of late modernity. As a matter of fact, in *CSI*, both killer and victim have been assigned new roles. In dramaturgical terms, they are only interesting as bearers of clues and evidence. This is the detectives' focus, which is why they have surprisingly little interest in the social dynamics that led to a violent crime in this or that case. Motives are essentially irrelevant, and the question of guilt is hardly ever posed. There is a dead body and a living one whose fateful meeting in the recent past generated forensically analyzable data. Along with the crime scene, these two bodies become a reservoir of traces, a pool of clues, a collection of crime-specific signifiers that are technically stabilizing and can be accessed at any time and related to one another. The art of recombining these data, whose origin, quality, and form may suggest a high degree of heterogeneity, is presented to viewers as a game of interpretation. Visually blurred sequences simulate a number of possible narratives. Each of these provisionally evaluative fragments shows the detectives where further clues might possibly be searched for and found. Data collection is extended and refined.

Metaphorically speaking, in *CSI* the world is treated as a database whose records must be mined and combined to gain insights into connections that are critical to the task at hand. With the data "called up" from the crime scene, the investigators generate "views"—in both the technical and cinematic senses—that enable them to draw conclusions about various relationships between things. This new mode of interpretative practice is obvious. It can be very dramatic, such as in the use of collagelike, abrupt cuts. It takes the discovery of many small details to finally piece together the narrative over the course of an episode. Each

member of the team contributes his or her own expertise and questions to the overall picture of a particular case; photographs, pathology, entomology, toxicology, materials science, information technology, and biotechnology know-how are called into play in the division of labor and appropriately linked. Even the arsenal of equipment and procedures, as it appears on the program's website, offers little in the way of a prefabricated connection. Rather, it represents a collection of individually retrievable insights, such as into various consequences for the lives of the characters and those of their actors and actresses. The visitor must form his or her own "views" of the relationships.

Clearly, traditional hermeneutics, which permitted the erstwhile television character Inspector Columbo (based on his personal intuition and legendarily battered notebook) to apprehend even the most artful malefactors by dint of persistent interpretation is now a thing of the past.[6] In the age of *CSI*, his approach, which emanated from the motive and sought to successfully interpret the incomplete explanations of the suspects, seems hopelessly antiquated. Columbo the seasoned professional, a super hermeneutist and psychologist rolled into one, who understood the author of a crime like nobody else, who empathized with the motive of the perpetrator, and who inevitably shared with viewers his correct interpretation, has been replaced by a team of scientific experts who proceed laboriously and systematically to sort through a plethora of data of varying quality and to combine them in a nonobvious way, first in tentative and then in increasingly more definitive simulations. Farewell to hermeneutics and psychology.[7]

As a mechanism and model, the database changed the way detectives do their work, leading, for example, to a higher number of solved cases per episode. The database also ensures that, on both sides of the screen, the combinatory degree of freedom of each "signifying practice" can be extended. Yet the cultural shift associated with *CSI* hardly occurs in the vacuum of the signifying game. When the most successful cultural-industrial products of an era present such basic communicative techniques as searching, interpreting, and understanding a mode of database querying, the question quite naturally arises about the relationship between database development and the transformation of the "signifying practices." This question is the subject of the following considerations.

The Historiographic Approach

The claim that the search-and-interpret culture of the late twentieth and early twenty-first centuries focused on the recombinatory potential

of computer-aided database technology must be examined in light of the history of that technology. Given the evolution that appears to have occurred between *Columbo* and *CSI*—that is, from the 1970s to the present—it would be helpful to keep in mind both how knowledge is disseminated and who disseminates it (the dramatis personae). The former includes the growing availability and combinatory potential of data as well as the greater freedom to address new, unanticipated questions to existing data. The latter refers to the functional differentiation of the various knowledge actors or the representation of the interpretative process as one of shared work. These major changes, in turn, can be seen as the consequence of the development of computer-aided database technology. This occurred in the 1970s and early 1980s and marked a period in which "flexibility" in several fields of social practice had a special appeal. From retailing to the military, from modes of transport to the technical universities, allocation of available resources was relaxed. To be able to react quickly to changing market conditions, adapt to expected opportunities for training, make optimal use of scarce resources, and both produce and distribute goods in accordance with the new principle of "just in time," knowledge of relationships and interdependencies took on enhanced significance. Operational concepts, information systems, and analytical techniques were changing at breathtaking speed. "[There] are no fundamental phenomena. There are only reciprocal relations, and the perpetual gaps between intentions in relation to one another," wrote Michel Foucault in 1982.[8]

In contrast to the televised crime scene investigation, when attempting to reconstruct the independent development of database culture and interpretative technology, it is not at all obvious where the clues and data crucial to assessing the links of interest may be found. Certainly, computer history has come far in recent years. Yet it is still dominated by studies that strive to package a highly complex sociotechnological transformation as a computer revolution. These studies share James Cortada's view that "more data faster" is the most important characteristic of this change.[9] Here, the question is simply whether the many exponential growth curves should be read as a sign of progress or of increasing threat.[10] On the other hand, three research trends that emerged in the last three years provide useful clues to understanding the relationship between database development and cultural transformation. The first of these trends is detailed investigation of user context found, for example, in the work of Paul Edwards.[11] A second trend is a departure from the primacy of hardware history in favor of software history, as proposed by Martin Campbell-Kelly.[12] Finally, a new history of computers

applies sociohistorical approaches to examining strategies for professionalization and governance of companies and administrations in the context of computer and software development. Examples include the work of Thomas Haigh on the early days of management information systems, Jon Agar's "government machine," and JoAnne Yates's study of the introduction of computers to the life insurance sector.[13] As Haigh, Agar, and Yates all agree, the origin and spread of machines can only be understood in association with various actors, artifacts, institutions, and discursive strategies.

Conditions for the unexamined history of relational databases are relatively favorable, since the popular standard narrative already suggests a fairly complex history of the development. Its central figure is Edgar F. "Ted" Codd, a misunderstood key theorist whose pioneering ideas were never really appreciated by IBM and made it to market only thanks to a competing academic project.[14] An obituary in the *New York Times* cogently summarized this history: "While working as a researcher at the I.B.M. San José Research Laboratory in the 1960's and 70's, Dr. Codd wrote several papers outlining his ideas. To his frustration, I.B.M. largely ignored his work, as the company was investing heavily at the time in commercializing a different type of database system." His insights were too "revolutionary" for IBM and, unlike the experience of others, also never made him rich: "It was not until 1978 that Frank T. Cary, then chairman and chief executive of I.B.M., ordered the company to build a product based on Dr. Codd's ideas. But I.B.M. was beaten to the market by Lawrence J. Ellison, a Silicon Valley entrepreneur, who used Dr. Codd's papers as the basis of a product around which he built a start-up company that has since become the Oracle Corporation."[15]

This account employs two popular motifs of invention and discovery stories: the ignorance of powerful market leaders and the suffering of sympathetic protagonists.[16] Such stories comprise a nifty genre for dealing with trials and tribulations that goes far beyond the normal scope of the historiography of computing, which focuses on rapid development. What this approach masks, however, is the fact that the challenges of developing a practical relational database model are directly connected to a major restructuring of the use of information. The story of "poor Ted Codd" not only overestimates the influence of an IBM CEO but also underestimates the tremendous changes taking place in how knowledge circulated and the massive transformation in the knowledge actor landscape, which development of relational databases both presupposed and brought about.

The New Separation of Powers

In 1970 the Association for Computing Machinery (ACM) published an article in its journal, *Communications*, that quickly became a key reference for the entire development community. In the article—titled "A Relational Model of Data for Large Shared Data Banks"—Ted Codd presented his ideas about a new database architecture.[17] Codd's article is still being cited today as a milestone, perhaps precisely owing to its abstract reasoning.[18]

The first sentence was already game-changing: "Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation)."[19] Codd assumed that the impending changes in the way knowledge circulated would make it necessary to redefine the competencies of the knowledge actors involved. The future operation of large databases would bring forth new user groups who had to be "protected" from having to know the internal organization and representation of the relevant knowledge stores.

There is, of course, much more to be read into such a first sentence. It implies, for instance, that the author actually knows the "future" and its "users" and that he has the power to define what "must" be done with them or what their relation to the machines they were to use should look like. Moreover, the sentence ignores predecessors and competitors who had been working on increasing the degrees of freedom of what they thought to be the "future users."[20]

Yet there must be a reason why such a prophetic statement or speech act would resonate at all with the general experience of its readers.[21] Codd's call for selective protection against knowledge was due to the fact that querying a complex database very quickly turned into a challenging task that required the sophisticated knowledge of a professional programmer to solve. Large databases were subject not just to constant change in the representation of data "as a result of changes in query, update, and report traffic and natural growth in the types of stored information." Both hierarchically constructed databases and databases with a network structure had the disadvantage that access pathways to stored information were predefined, and new queries were forced along these pathways. Without precise knowledge of the trajectories, new queries could not be executed. Codd promised that a relational database structure would serve a substantially larger group of users who might be naive about information technology but who knew how to query: "Activities of users at terminals and most application programs should remain unaffected when the internal representation of data is

changed and even when some aspects of the external representation are changed."[22] The fact that they no longer had to worry about the form in which data were stored would free users to concentrate more on flexible retrieval. No longer would they have to rely on experts. Now, programmers would be responsible for providing reliable supply and availability (i.e., for the organization and representation of data) and consequently protected from the increasingly specialized query needs of database users. Thus, in 1970—long before Larry Ellison—nobody noticed that, on the user side of databases, a transition from the targeted searching of stored data to the generalized querying of data banks (i.e., a transition from research to interrogating the oracle) had taken place.[23]

Nevertheless, the future division of labor and power foreseen by Codd was linked to changes in software architecture, which initially could only be theoretically estimated and mathematically formulated. What was clear, at least, was that the database of the future would have to be so configured that its contents could also deal with questions that had not yet come up during the planning of the database. Codd knew what had to be done in order to achieve this goal. It was necessary first to develop a simple and general organization of data in tables that could be linked together by keys; second, to create a management tool to facilitate constantly changing records and expansion of the database; and third, to establish a practical query language that satisfied mathematical requirements yet was close to the natural language of users with no knowledge of programming. That this list of tasks occupied more than one of Codd's articles goes without saying. Just clarifying the model—its advantages and disadvantages as well as differences among competing designs—took several years of intense debate to complete.[24]

The more concrete the description of Codd's model, the greater the uncertainty among database specialists familiar with all the details of conventional models. The confusion arising out of so many different system designs being proposed simultaneously, and the future scenarios involving them, was immense.[25] Rhetorical claims about performance and flexibility primarily served for demarcations between two camps either dealing with the same problems and proposing different solutions, or solving different problems with similar intentions. Those who both wished to preserve the privileged role of the programmer or the yet unstable figure of a database administrator, and who saw them as professional navigators through the treacherous waters of complex data stores, were especially confused and unsettled.[26] One of the most unsettled among them, Edgar H. Sibley, tried to bring order to the navigators' world by commenting and classifying the similarities and differences among the various designs:

"Practitioners of database technology have been somewhat confused by the many different systems for describing and manipulating data. The two major approaches that have emerged may be termed the relational or set theoretic, and the data structured or procedural. There are obviously differences in these, but there are also similarities."[27]

The sorting capability of Sibley's lower-level "procedural" language did not stop the "relationalists" from pouring more oil onto the fire. Examples included a paper by Codd and Christopher J. Date titled "Interactive Support for Non-Programmers: The Relational and Network Approaches," as well as another publication jointly authored by the two experts that purported simply to review then-current concepts but in actuality emphasized the specific advantages and the dramatic consequences of their own model.[28] That paper stated: "It may be argued, in fact, that the relational model is a representation of the data in terms of its natural structure only—it contains absolutely no consideration of storage/access details (pointers, physical ordering, indexing, or similar access techniques . . .); in a word, no 'representation clutter.'"[29] Date and Codd obviously intended such statements not only to show the mathematical and software-health benefits of their relational model but also to needle database specialists, for whom pointers, indexing, and physical ordering hardly constituted "representation clutter"; rather, they were the guarantors of efficient database querying.

In a second line of rhetorical scrimmage, Date and Codd highlighted the advantages for the technically naive user, that is, for housewives and managers, thus suggesting that the broader user markets of the future could only be opened and served by relational databases: "The relational model is a particularly suitable structure for the truly casual user (i.e., a non-technical person who merely wishes to interrogate the database, for example, a housewife who wants to make enquiries about this week's best buys at the supermarket). In the not too distant future the majority of computer users will probably be at this level."[30] For a database specialist of the early 1970s who had devoted all his programming art into building a resource-saving, efficient query technology, this was a nightmare scenario. Yet Codd and Date proceeded to align the role of the future programmer with the "casual user," thus forcing the programmer to accept the consequences of strict "data independence" and "user independence" that followed from it: "The system must therefore be capable of supporting such a view at the casual user level—and this in itself is a strong argument for providing the same view at the programmer level, since any operation at the casual user level must be implementable at the programmer level too."[31]

The debate reached a climax during a panel discussion titled "Data Models: Data-Structure Set vs. Relational," to which the counterparties were summoned in the context of a conference organized by the ACM in 1974 and for which some of the articles quoted above had been prepared.[32] Codd's most prominent opponent was Charles W. Bachman, who had been awarded the Turing Award the year before.[33] The account of their "great debate" reflects an extremely tense discussion that at every instant threatened to spiral out of control. In his second set of remarks, Codd made clear how enormous the differences between them were: "With the relational model we have well-known high-level logics to act as the target temporarily to get the query accurately and precisely defined—whereas with networks, as far as I am aware, there are no comparable logics available at that level to pin down the semantics of the casual user's transactions. I would like to be corrected if I am wrong." He was subsequently corrected so often that, toward the end of the debate, he even had to defend against the accusation that his relational stance was taking on religious overtones. Indeed, the debate between Codd and Bachman revealed such fundamental differences that the mutual suspicion of fundamentalism was not easily dismissed. The discussions continued to be characterized by mistrust and mutual skepticism. Even when Codd attempted to articulate the differences a little more carefully and to inject a more conciliatory tone to the controversy, Bachman still believed he was being snookered. "I think we have just seen the red herring that we want to avoid," he retorted to Codd's ostensibly matter-of-fact question as to whether the data structure should contain some form of storage and access information or not. "There are all kinds of logical and arithmetic semantic constraints that you may want to impose on a database," Codd conceded. "I think it is a debatable point as to whether some of these should be in the data structure and some in separate declarations." The deciding question for him was only how to do it: "You cannot devise in advance a set of data types that will give you all the possible semantic checks that any database could require. The issue is whether you should put some of the semantic constraints in the data structure of your principal schema and some elsewhere, or all elsewhere."[34]

That Codd's polemic assumed such subtle forms failed to improve the atmosphere of the discussion, especially since he let pass few opportunities to score points. Digs such as "I am rather surprised by your military, that is, hierarchical approach to security" were not infrequent. Again and again, the rights of users were pitted against the dominance of programmers. A case in point is Codd's attack on the programmer

focus evinced by the Database Task Group of the Conference on Data Systems Languages (CODASYL): "I am at a loss to understand why there has not been more emphasis during the past six years by CODASYL on users other than the application programmer."[35] When Edgar Sibley somewhat lamely parried this accusation with the claim that CODASYL did have an End Users Facilities Task Group, which had been up and running for four months, Codd shot back: "The fact that this group has been established so recently supports what I just said."[36]

The debate ended inconclusively. All that was certain was that Bachman and his followers were committed to a goal-oriented, well-controlled use of technical resources and Codd and company. to the widest possible interpretative independence for an increasingly heterogeneous user community. The question of limiting and expanding access to databases, the relationship between the programmer's authority to control and the flexible querying power by users of the world as database, and the trade-off between technical performance and mathematic elegance were additional themes that helped to sharpen the battle lines between the two camps. Bachman himself, in a long interview with Thomas Haigh in 2004, showed that he had not changed his attitude toward relational databases (asserting, "I'm still skeptical"), nor did he grant that Codd was right, even in light of the success of his concept. Rather, Bachman pointed out that each of the warring parties had been backed by powerful competitors: IBM on Codd's side, and Honeywell on Bachman's.[37]

The implication of Bachman's argument was that, on the one hand, the relational model was supported by companies oriented to the service sector, whereas the procedurally oriented model was promoted by industrially mixed companies dominated by chemistry, defense, and process automation. Yet this company history explanation is not convincing. First, both IBM and Honeywell offered their customers successful procedural database systems: an evolution of the CODASYL-based network database system IDS (Honeywell), and the hierarchically structured IMS system introduced in 1968 (IBM).[38] Honeywell and IBM were jousting in the same market for customers; both were thinking along the lines of a relational database but somewhere in the future. Second, with its 1976 Multics Relational Data Store (MRDS), Bachman's employer, Honeywell, had already developed the first commercially released relational database management system, but only when packaged with the Multics Integrated Data Store, which was based on the CODASYL model.[39] Third, both Bachman and Codd had reasons for dissociating themselves from their employers' current products, not

only with respect to the future design of databases. Bachman had undergone his professional socialization not at Honeywell but in the 1950s at Dow Chemical and in the 1960s at General Electric. At the time of the great debate, he had only been active at Honeywell for four years due to the fact that Honeywell purchased General Electric's entire computer business. In 1980 he left the company.[40] Codd, in contrast, was an IBM man through and through. But the distance between research and corporate at IBM, along with the story widely circulated by Codd that IBM had given him and his field too little support, did not augur well for an unconditional identification with the company's concerns.[41]

The contrasts between the two camps were reflected in the completely different career paths of the two protagonists. At General Electric, Bachman had already had a hand in shaping the transition of an industrial plant to the computer age. His interests involved functioning systems and practical problem solving. For him, cost-effectiveness was a top priority in operating any data center. That, in turn, meant no less than operating the databases so that they could interact as seamlessly as possible with the application programs. Consequently, programmers had to be closely familiar with both data and applications and to be versed in countless tricks for keeping the precisely tuned interactions from collapsing.[42]

In fact, Bachman and Codd do not represent different corporate identities. Rather, they embody the cultural contrast between the technologist and the academic, or between the applied scientist and the theoretician. Their cultural imprints and images of themselves could not have been more different. Especially at odds were their ideas about who would deal with databases in the future, including who would build them, maintain them, oversee them, and use them. This difference reflected the relationship of users to machines. Whereas technically savvy users could tinker with their own machines, technically inexperienced users were instructed to treat their machines as black boxes. Machines used by "geek" types could always be improved and expanded. Systems targeted to managers and other "casual" users had to be reliable. Whereas Bachman was strongly in favor of specialized, optimizable database machines operated by properly trained experts, Codd envisaged a future user's desire for a solidly built but multifunctional database system.

For his part, Ted Codd was a British mathematician who, after serving in the Royal Air Force during World War II, took a job with IBM in New York. During the McCarthy era, he immigrated to Canada. On his return, he studied at the University of Michigan, where he received his

PhD, and then in 1967 shifted to the protected work environment of the IBM Research Laboratory in San José, California. For Codd, a database had to be mathematically describable and the corresponding theory simple and elegant.[43]

Relationality in the Bay Area

By the mid-1970s the central concepts of Codd's model had been clarified and made reliably available. The characteristics of a relational database were presented by Morton Astrahan and Donald Chamberlin as follows: "In a series of papers, Codd has introduced the relational model of data and discussed its advantages in terms of simplicity, symmetry, data independence, and semantic completeness."[44] All the data in a relational database system had to be capable of being represented in a matched set of clearly labeled tables, so-called relations. Each relation comprised a set of clearly labeled tuples and attributes. The ordering of tuples was irrelevant, but each tuple represented an addressable part of the entity described by the relation. It had to be distinct from other parts of that entity and could only occur once. In addition, each relation was assigned an attribute that served as the primary key.[45] Theoretically, what "data independence" signified had been made clear in the discussions of the preceding years, especially in the debate with the adherents of procedural database architecture. However, what the objective of enhanced "user independence" really required emerged only once development was actually under way.

Between 1974 and 1979 the IBM Research Laboratory in San José, California, instituted a project later known as System R.[46] An initial phase that ran from 1974 to 1975 produced the Structured English Query Language (SEQUEL, later SQL), which would enable future users to formulate their queries at an interactive terminal.[47] Great importance was attached to the "human factor," and many experimental studies were carried out on learnability and the friendliness of the system. In the second project phase (1976–77), the system was reconfigured for multiple users working simultaneously. Moreover, the existing SQL was adapted for use with a variety of different systems. Users familiar with the programming languages PL/I and Cobol would have the same benefits and could use the same syntax as so-called ad hoc query users.[48] In 1978–79 operating tests were conducted within the company and with three customers, and the user experiences were evaluated.

No doubt, System R was a major experiment for IBM that demonstrated user response to the system. The creation of "user friendly

interfaces" and a major modularization of the system were the most important strategies for reaching the project objectives.[49] Work on data and user independence, however, proved a massive headache for the developers. Whereas the second version of the query language appeared to function very well, the System R developers continued to struggle with the question of how to structure the data without burdening users with technical details. The problem that Bachman found central—that of how to anticipate the huge storage and processing capacity required to operate a relational database—also seemed insurmountable. A step toward a solution came in the form of an idea put forward by Rudolf Bayer and Edward M. McCreight back in 1972. B-trees were to help more efficiently index stored data. Query read-and-write operations could be reduced by an order of magnitude without having to couple the data structure to the query.[50]

The story of Ted Codd holds that IBM listened to him either too late or not at all, and therefore the development of a marketable relational database system was delayed by years. In fact, IBM was investigating Codd's ideas on a much broader front than Codd himself used to acknowledge, such as in the case of the creation of the Peterlee Relational Test Vehicle (PRTV).[51] Likewise, a critical reading of the history of System R affords another interpretation. Sure, during the development of System R, the experimental status of the project was constantly emphasized, and the project was always referred to as a "feasibility study" in that "at each user site, System R was installed for experimental purposes only, and not as a supported commercial product."[52] Nevertheless, the project had a considerable number of employees and was carried out by a large, multifunctional group.[53] Within IBM, System R was much more than the hobby of a few engineers. Particularly striking is the fact that IBM published very detailed descriptions of the progress of System R for the technology community.[54] That could only mean that the status of System R within IBM was more respected than Codd was willing to admit. Because the system as yet had no strategic (i.e., commercial) significance and thus was not proprietary, it was free to seek support outside.[55] System R publications were invitations to collaborate. The popular theory that it was primarily the success of the University of California, Berkeley, project INGRES, which "finally" forced IBM to act, has to be rethought.[56] System R developers openly sought collaborators in other development centers outside IBM and from 1975 onward actually found them locally, namely, at UC Berkeley, where since 1973 Michael Stonebraker and Eugene Wong had been leading an effort to build a prototype relational database system.[57] It is no surprise that

academic computer scientists had accepted Codd's ideas, as even in the early 1970s they had percolated into mathematically interesting, precommercial fields that justified academic activity.

The circumstances of the two projects in Berkeley and San José could not have been more different: academic freedom on the one side, professional project management culture on the other. In Berkeley, a modest PDP-11 computer operated somewhat shakily on UNIX, while in San José a powerful IBM 370 mainframe was directed by a variety of highly reliable operating systems.[58] A constantly changing student workforce at Berkeley—"a collection of goofy academicians"—contrasted with the controlled working conditions of experienced programmers and project managers in San José.[59] Stonebraker's conspicuously ironic portrayal of the INGRES project at Berkeley is in stark contrast to the sober publication style of the IBM engineers working on System R. In reading Stonebraker's reports over the course of his project, it becomes clear that INGRES was up against far greater challenges than IBM's System R. Simply procuring a computer and research funds, not to mention the constant turnover of student workers, complicated Stonebraker and Wong's task. Yet the competitive pressure between IBM and the university was evident in a trip Codd made to Berkeley to see exactly what Stonebraker and his team were up to. Stonebraker later recalled the IBM guru's visit with horror: "In January 1975 we invited Ted Codd to come to Berkeley in early March to see a demonstration of INGRES. The final two weeks before his visit everyone worked night and day so that we would have something to show him. What we demonstrated was a very 'buggy' system."[60]

Although the teams discussed the same topics, they very often came to different conclusions. Thus, to the question of whether static or dynamic storage procedures should be applied, the dynamic Berkeley team paradoxically answered "static" (later recognized as a mistake), whereas the somewhat stodgy IBM developers preferred the B-tree structure while working on the System R research project. Neither group decided lightly, and their arguments, for example, for and against the use of B-trees were published in a series of papers.[61] Nonetheless, the experients and experimental designs of both teams increased the collective experience of the "relational" camp among the community of virtual developers.[62]

A closer look at the two projects, however, reveals a whole range of commonalities. Apart from the fact that both teams were driven by the theoretical fascination of the relational model—Codd as the dogmatic point of reference was controversial in both places—INGRES and

System R poured enormous intellectual resources into their projects.[63] However different their ways and conditions of working, their efforts continued to advance and were mutually productive. Both groups made serious strategic miscalculations, taking risky shortcuts that they later regretted and exorcised in public reports. That INGRES and System R promised more than they could ever deliver and that their leaders later claimed to have achieved more than they did is probably an artifact of the discourse peculiar to projects and reports. What Stonebraker stated regarding INGRES was also true of System R: "Our goals expanded several times (always when we were in danger of achieving the previous collection)."[64]

Both teams devoted substantial effort to developing a "high-level search and query language."[65] This was crucially important for the future of the relational model because the user friendliness of the language would determine expansion into the nontechnical user community. With the QUEL query language developed at Berkeley and ALPHA, SQUARE, SEQUEL, and SQL all created at San José, developers pursued their key objective of "user" and "data" independence and placed a power tool in the hands of a new type of database user. By far the most convincing example for the unprecedented power of both a "high level search and query language" in particular and the relational database in general was to be found in a 1976 article by Stonebraker from a time in which the work on relationality was going full speed across the Bay Area. Stonebraker's example was the following: Imagine a company that can be described by two relations. The first relation or table, "EMP," is one typically seen in the personnel department of a company and contains data representing an employee's name, department, salary, superiors, and age. A second relation or table, "DEPT," represents the space allocation department and contains the allocation of departments to the various floors in the company building. The two relations are represented as follows:

EMP (NAME, DEPT, SALARY, MANAGER, AGE)
DEPT (DEPT, FLOOR#)

A three-line QUEL query by a manager tasked with restructuring the company might look like this:

```
RANGE OF E IS EMP
RANGE OF D IS DEPT
DELETE E WHERE E.DEPT = D.DEPT AND D.FLOOR# = 1
```

Such a query changes the EMP relation and, correspondingly, an employee relation. As a guide to action, this operation is equivalent to the following plain text command: "Fire everybody on the first floor."[66] The cynical empathy that the INGRES team's use of this example showed for the role of a manager is hard to top. Slightly more sensitive was the query that the System R team published a year earlier, written in SEQUEL:

```
SELECT NAME
FROM E IN EMP
WHERE SAL >
   SELECT SAL
   FROM EMP
   WHERE MNO = E.MGR;;
```

The corresponding expression in management-speak was: "Find names of employees who earn more than their manager."[67]

Both questions are aimed at a new database user in the hope of persuading him or her of the appeal of a powerful query ability and showing how relational databases can rapidly respond to questions whose answers require the use of links that the question never anticipated. That this capability to interrogate data already collected by a company was relevant to managers went without saying. Since the good old days of scientific management, managers had been required to apply the most refined information technology skills to discover or to construct links across the corporate organizational structure and that could not be detected in dedicated tables.

Perspectives on Relationality

Work on the relational database revived old hopes. As early as the 1960s, corporate advisers had anticipated a powerful "management information system" in the near future. Formerly scattered reports, forms, and memoranda would all be gathered together in a single pot. Some even dreamed of a future "electronic data bank, or pool of information, from which reports of many types can be drawn."[68] These systems would not require managers to attend computer or programming courses. As Alfred Eisenpreis noted, "Computers are useful, but only if we can ask them questions that matter. It is the task of management and science to help us to clarify these questions."[69] By the mid-1970s, thanks to advances in relational databases, the vision of the computer as a powerful management instrument that had only to ask intelligent questions of a

vast reservoir of data appeared to be nearing reality. In the foreseeable future, managers would be able to "read" their company like an open book.[70]

The computer evolved from an instrument for making rapid calculations to a tool for restructuring. The diversity of relationships enabled by the computerized database system not only increased the data-processing capacity of a company. By applying the new combinatorial capability offered by relational databases, computers now promised to reduce a firm's internal transaction costs. Even middle management would soon be able to make cost comparisons by cross-querying tables and departments. That this possibility materialized at a time (the late 1970s and early 1980s) when the restructuring of entire companies was a daily occurrence dramatically enhanced the appeal of relational database technology, especially when Oracle implemented relational database systems on smaller computers.

The increase in the independent interpretation of business management or, more generally, the growing interpretative freedom of database users also coincided with a fundamental questioning of the meaning of interpretation and the rules governing it. For example, in the 1970s this issue was discussed extensively in relation to fundamental problems with the interpretation of texts. Here, of course, the "user" was rather called the "reader." As Roland Barthes wrote in *S/Z* (1970), "To interpret a text is not to give it a (more or less justified, more or less free) meaning, but on the contrary to appreciate what *plural* constitutes it."[71] For Barthes, a text is a "galaxy of signifiers" in an infinitely complex and varied network of interactions. Interpretation thus does not imply trying to elicit the author's original meaning through hermeneutically approved interrogation techniques. Rather, as Umberto Eco once put it, a text is a machine for eliciting interpretations.[72] The separation between author and reader implied by the concept of text is as strict as that separating programmers and users. To quote Barthes again, "We gain access to [the text] by several entrances, none of which can be authoritatively declared to be the main one."[73]

Umberto Eco's *The Open Work* (1962) and Susan Sontag's "Against Interpretation" (1964) represented the first steps toward a critique of the hermeneutic process. By 1970 even the comfort of Hans-Georg Gadamer's *Wahrheit und Methode* (Truth and method) could not stop the trend—a common, meaningful, cultural unity of authors and readers was launched.[74] Whereas Michel Foucault on occasion raised the question of what constitutes an author as a way of rethinking analytical discourse and classifying the expressible and Barthes proclaimed

the death of the author, Wolfgang Iser wrote of "indeterminacy as the condition of literary prose."[75] The notion of the emancipated and independent reader, now freer to interact with his or her text than ever before, was all the rage. This notion was supported by the fluidity of any interpretative work, namely, the options for presenting the text. Hence Dieter E. Sattler's exhortation that even classical texts should be printed in such a way as to support the interpretative autonomy of the reader rather than discourage it.[76] Metaphorically speaking, in so doing he changed the structure of the database underpinning the act of interpretation. In response to the (highly political) dispute over the "correct" understanding of Friedrich Hölderlin's manuscripts, Sattler's publication of the Frankfurt edition was strong on philological and typographical innovation. The edition was intended to enable a flexible combination of interpretative elements, since the "needs of readers had become more critical, independent, and expressive."[77] Only such a treatment would leave each layer of text and the relationships between parts of the text free, giving the reader sufficient flexibility in the *ars combinatoria* of his or her reading or questioning of the text.

"In its advocacy of artistic structures that demand a particular involvement on the part of the audience, contemporary poetics merely reflects our culture's attraction for the 'indeterminate,' for all those processes which, instead of relying on a univocal necessary sequence of events, prefer to disclose a field of possibilities, to create 'ambiguous' situations open to all sorts of operative choices and interpretations," writes Umberto Eco in *The Open Work*.[78]

The 1970s showed that texts were like databases. Both had undergone a theoretical and practical reconfiguration, and both allowed reconstruction of the available material in a number of ways. Each could be understood as an ambiguous field of possibility amenable to different operative and interpretative procedures and decisions. Their presentation should not be such as to limit or prestructure their interpretation, nor should they speak for themselves. Texts share commonalities with electronic, biological, and materials databases that constitute the world of the forensic investigators in *CSI*. Links can only be simulated, checked, and recognized by querying informational fields of possibility. Accordingly, they require special technologies, procedures, and languages that generate new meaning from existing data. Remember the *CSI* laboratory chief's answer when asked why he worked as a forensic investigator: "Because the dead can't speak for themselves."[79] Neither, we must conclude, can data.

Notes

I thank Daniela Zetti and Lea Haller for logistical support and critical comments, which were of enormous help in writing this article. I am very much indebted to Giselle Weiss, who managed to translate the German version of this article. For the original work, see David Gugerli, "Die Welt als Datenbank: Zur Relation von Softwareentwicklung, Abfragetechnik und Deutungsautonomie," in *Daten*, ed. David Gugerli et al. (Nach Feierabend: Zürcher Jahrbuch für Wissensgeschichte, Zürich/Berlin, diaphanes. 3), 11–36.

1. Guy Adams, "CSI: The Cop Show That Conquered the World," *Independent*, December 19, 2006.

2. Mikael Krogerus, "Der Serientäter: Interview mit Steven Bochco," *NZZ Folio*, no. 10 (2006). See also Leif Kramp and Stephan Alexander Weichert, "Im Schatten des CSI-Effekts: TVSerienverdrängen den Kinofilm," *Medienheft*, January 22, 2007.

3. Nicholas J. Schweitzer and Michael J. Saks, "The CSI Effect: Popular Fiction about Forensic Science Affects Public Expectations about Real Forensic Science," *Jurimetrics* 47 (2007): 1–8; Michael D. Mann, "The 'CSI Effect': Better Jurors through Television and Science?," *Buffalo Public Interest Law Journal* 24 (2006): 157–83; Donald E. Shelton, Gregg Barak, and Young S. Kim, "A Study of Juror Expectations and Demands Concerning Scientific Evidence: Does the 'CSI Effect' Exist?," *Vanderbilt Journal of Entertainment & Technology Law* 9 (2006): 331–68; Tom R. Tyler, "Viewing CSI and the Threshold of Guilt: Managing Truth and Justice in Reality and Fiction," *Yale Law Journal* 115 (2006): 1050–85; Kimberlianne Podlas, "The C.S.I. Effect: Exposing the Media Myth," *Media and Entertainment Law Journal* 16 (2006): 429–65.

4. Apart from CSI and its already cited derivatives, for example, *Without a Trace* (2002), *Cold Case* (2003), *NCIS* (2003), *Criminal Minds* (2005), and *Shark* (2006).

5. An interview by Norbert Schneider in the *Frankfurter Allgemeine Zeitung* is an exception: "Fernsehserie 'CSI': Der Mensch als toter Körper," *Frankfurter Allgemeine Zeitung*, February 21, 2006, 48.

6. That Columbo still has a fan base is, of course, uncontested. See Armin Block and Stefan Fuchs, *Columbo: Das große Buch für Fans* (Berlin, 1998).

7. Of course, this farewell to hermeneutics does not mean that hermeneutics has completely and forever disappeared from the screens and the minds of our time.

8. Michel Foucault, "Espace, savoir et pouvoir," in *Dits et écrits 1954–1988 par Michel Foucault*, ed. Daniel Defert and François Ewald (Paris: Éditions Gallimard, 1994), 270–85.

9. James W. Cortada, *The Digital Hand: How Computers Changed the Work of American Manufacturing, Transportation, and Retail Industries* (New York: Oxford University Press, 2003); James W. Cortada, *The Digital Hand: Volume II: How Computers Changed the Work of American Financial, Telecommunications, Media, and Entertainment Industries* (New York: Oxford University Press, 2005).

10. Gene I. Rochlin, *Trapped in the Net: The Unanticipated Consequences of Computerization* (Princeton, NJ: Princeton University Press, 1998).

11. Paul N. Edwards, "From 'Impact' to Social Process: Computers in Society and Culture," in *Handbook of Science and Technology Studies*, ed. Sheila Jasanoff

et al. (Thousand Oaks, CA: SAGE Publications, 1994), 257–85; Paul N. Edwards, *The Closed World: Computers and the Politics of Discourse in Cold War America* (Cambridge, MA: MIT Press, 1996).

12. Paul E. Ceruzzi, *A History of Modern Computing* (Cambridge, MA: MIT Press, 1998); Martin Campbell-Kelly, *From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry* (Cambridge, MA: MIT Press, 2003). See also Ulf Hashagen et al., eds., *History of Computing: Software Issues* (Berlin: Springer, 2002).

13. Thomas Haigh, "Inventing Information Systems: The Systems Men and the Computer, 1950–1968," *Business History Review* 75 (2001): 15–61; Jon Agar, *The Government Machine: A Revolutionary History of the Computer* (Cambridge, MA: MIT Press, 2003); JoAnne Yates, *Structuring the Information Age: Life Insurance and Technology in the Twentieth Century* (Baltimore, MD: Johns Hopkins University Press, 2005).

14. United States National Research Council, ed., *Funding a Revolution: Government Support for Computing Research* (Washington, DC: National Academies Press, 1999), 159–68.

15. Katie Hafner, "Edgar Codd, Key Theorist of Databases, Dies at 79," *New York Times*, April 23, 2003.

16. See, for example, the account of the history of Web browser technology and the role of Marc Andreesen in Robert H. Reid, *Architects of the Web: 1000 Days That Built the Future of Business* (New York: John Wiley & Sons, 1997).

17. Edgar F. Codd, "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM* 13 (1970): 377–87.

18. The ACM Digital Library lists 799 articles citing Codd. See http://portal.acm.org/.

19. Codd, "Relational Model," 377.

20. For an earlier concept of an end user database, see the stream of work at the System Development Corporation in the 1960s, Charles P. Bourne and Trudi Bellardo Hahn, *A History of Online Information Services: 1963–1976* (Cambridge, MA: MIT Press, 2003).

21. "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" (oral history of C. J. Date, interviewed by Thomas Haigh, Computer History Reference number X4090.2007, 13–14, http://archive.computerhistory.org/resources/access/text/Oral_History/102658166.05.01.acc.pdf).

22. Codd, "Relational Model."

23. On Larry Ellison and the history of Oracle, see Matthew Symonds, *Softwar: An Intimate Portrait of Larry Ellison and Oracle* (New York: Simon & Schuster, 2003).

24. See Donald D. Chamberlin et al., "A History and Evaluation of System R," *Communications of the ACM* 24, no. 10 (1981): 632–46.

25. Gordon C. Everest, "The Futures of Database Management," *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access, and Control*, May 1–3, 1974, 445–62.

26. Charles W. Bachman, "The Programmer as Navigator," *Communications of the ACM* 16 (1973): 653–58.

27. Edgar H. Sibley, "On the Equivalences of Data Based Systems," *Proceedings of the 1974 ACM SIGFIDET (now SIGMOD) Workshop on Data Description, Access, and Control. Data Models: Data-Structure-Set versus Relational*, May 1–3, 1975, 43–76.

28. Edgar F. Codd and Christopher J. Date, "Interactive Support for Non-Programmers: The Relational and Network Approaches," *Proceedings of the 1974 ACM SIGFIDET. Data Models*, 11–41.

29. Christopher J. Date and Edgar F. Codd, "The Relational and Network Approaches: Comparison of the Application Programming Interfaces," in *Proceedings of the 1974 ACM SIGFIDET. Data Models*, 95.

30. Ibid.

31. "Data independence" was defined by Date in 1977 as "immunity of applications to change in storage structure and access strategy" (Christopher J. Date, *An Introduction to Database Systems* [Boston: Addison-Wesley, 1977], cited in Chamberlin et al., "History and Evaluation," 632). Text quote from Date and Codd, "Relational and Network Approaches," 95.

32. Everest, "Futures of Database Management"; Codd and Date, "Interactive Support"; Date and Codd, "Relational and Network Approaches"; Sibley, "On the Equivalences."

33. Thomas Haigh, "Charles W. Bachman: Database Software Pioneer," *IEEE Annals of the History of Computing* 33, no. 4 (2011): 70–80. The Turing Award is given annually by the Association of Computing Machinery (ACM) "to an individual selected for contributions of a technical nature made to the computing community. The contributions should be of lasting and major technical importance to the computer field" (http://awards.acm.org/homepage.cfm?awd=140).

34. Panel and Audience, "Discussion," in *Proceedings of the 1974 ACM SIGFIDET. Data Models*, 123, 134, 128.

35. Ibid., 137, 132. In the context of CODASYL, a debate had been ongoing since 1959 over programming languages that could be used on a range of computers regardless of the manufacturer. In 1969 the Data Base Task Group published the first description of a language for the network database model. Vis-à-vis the work on CODASYL at the time of the debate between Codd and Bachman, see "CODASYL Data Description Language Committee: CODASYL Data Description Language," *Journal of Development* (1973). See also T. William Olle, *The Codasyl Approach to Data Base Management* (Hoboken, NJ: John Wiley & Sons, 1978); and J. S. Knowles and D. M. R. Bell, "The Codasyl Model," in *Databases—Role and Structure: An Advanced Course*, ed. Peter M. Stocker et al. (Cambridge: Cambridge University Press, 1984).

36. Panel and Audience, "Discussion," 132.

37. Charles W. Bachman, interview by Thomas Haigh, September 25–26, 2004, Tucson, Arizona. Interview conducted for the Special Interest Group on the Management of Data (SIGMOD) of the Association for Computing Machinery (ACM). Transcript and original tapes donated to the Charles Babbage Institute. *ACM Oral History Interviews* (2006): 104.

38. Thomas J. Bergin and Thomas Haigh, "The Commercialization of Data Base Management Software 1969–85," *IEEE Annals of the History of Computing* 31, no. 4 (2009): 26–41. See also Thomas Haigh, "How Data Got Its Base: Generalized Information Storage Software in the 1950s and 1960s," *IEEE Annals of the History of Computing* 31, no. 4 (2009): 6–25.

39. James A. Weeldreyer and Oris D. Friesen, "Multics Relational Data Store: An Implementation of a Relational Data Base Manager," in *Eleventh Hawaii International Conference on Systems Sciences*, January 5–6, 1978, 52–66.

40. See Bachman, interview, 2.

41. J. N. Gray et al., "Granularity of Locks in a Large Shared Database," in *Proceedings of the 1st International Conference on Very Large Data Bases, Framingham, Mass., ACM, 1975* (Framingham, MA: ACM, 1975): "The ideas presented here were developed in the process of designing and implementing an experimental data base system at the IBM San Jose Research Laboratory. (We wish to emphasize that this system is a vehicle for research in data base architecture, and does not indicate plans for future IBM products)" (447).

42. Bachman, interview, 19; Haigh, "Inventing Information Systems."

43. Christopher J. Date, "Edgar F. Codd. August 23rd, 1923–April 18th, 2003," *SIGMOD Record* 32 (2003): 4–13.

44. Morton M. Astrahan and Donald D. Chamberlin, "Implementation of a Structured English Query Language," *Communications of the ACM* 18 (1975): 580. See also Codd, "Relational Model"; Edgar F. Codd, "Relational Completeness of Data Base Sublanguages," in *Data Base Systems*, ed. Courant Computer Science Symposia (Englewood Cliffs, NJ, 1971), 65–98; Edgar F. Codd, "A Data Base Sublanguage Founded on the Relational Calculus," in *1971 ACM SIGFIDET Workshop*, 1971, 35–68.

45. Astrahan and Chamberlin, "Implementation," 580.

46. The description is based on Chamberlin et al., "History and Evaluation."

47. Astrahan and Chamberlin, "Implementation."

48. Chamberlin et al., "History and Evaluation," 636.

49. Ibid. On IBM's internal definition of "user orientation," see the corresponding entry in the *IBM Jargon and General Computing Dictionary*: "user orientation n. A term for the kind of manual now described as user friendly. This was first used in 1968 at a meeting of hardware publications managers, when it was stated that the greatest need was for publications developers to understand who the readers were and why they were readers. It took fifteen years for this truth to be widely appreciated and applied" (Mike Cowlishaw, *IBM Jargon and General Computing Dictionary* [Winchester, UK, 1990], 57). The (ex post) stated objective of the project was, in part, as follows: "(1) To provide a high-level, nonnavigational user interface for maximum user productivity and data independence. . . . (3) To support a rapidly changing database environment, in which tables, indexes, views, transactions, and other objects could easily be added to and removed from the database without stopping the system. . . . (6) To provide a flexible mechanism whereby different views of stored data can be defined and various users can be authorized to query and update these views. (7) To support all of the above functions with a level of performance comparable to existing lower-function database systems" (Chamberlin et al., "History and Evaluation," 633).

50. Rudolf Bayer and Edward McCreight, "Organization and Maintenance of Large Ordered Indexes," *Acta Informatica* 1 (1972): 173–89; "Symmetric Binary B-Trees: Data Structure and Maintenance Algorithms," *Acta Informatica* 1 (1972): 290–306.

51. Stephen Todd, "The Peterlee Relational Test Vehicle—a System Overview," *IBM Systems Journal* 15, no. 4 (1976): 285–308.

52. Chamberlin et al., "History and Evaluation," 634.

53. Paul McJones lists around forty-five articles at "Bibliography of the System R Project," http://www.mcjones.org/System_R/bib.html (December 4, 2000).

54. Paul McJones has compiled an incomplete list of the most prominent developers at "(Some) people who worked on System R and related systems," http://www.mcjones.org/System_R/people.html.

55. IBM development projects were strikingly different in terms of prestige, resources, and urgency. Representative labels included Hobby, Adtech, and Strategic Project. In the mid-1970s System R was classified under Adtech. In the *IBM Jargon and General Computing Dictionary*, Adtech was described as "Advanced Technology. Time put aside for a risky project, not necessarily directly related to a product. May mean: a) Play time (when someone else is doing it), or b) Exciting, innovative system design with no product deadlines (when speaker is doing it)" (Cowlishaw, *IBM Jargon and General Computing Dictionary*, 3).

56. United States National Research Council, *Funding a Revolution*, 162–68. INGRES stands for Interactive Graphics and Retrieval System. Michael Stonebraker et al., "The Design and Implementation of INGRES," *ACM Transactions on Database Systems (TODS)* 1 (1976): 189–222.

57. Michael Stonebraker, "Retrospection on a Database System," *ACM Transactions on Database Systems* 5 (1980): 225–40, 228.

58. "In order to provide a useful host-language capability, it was decided that System R should support both PL/I and Cobol application programs as well as a standalone query interface, and that the system should run under either the VM/CMS or MVS/TSO operating system environment" (Chamberlin et al., "History and Evaluation," 636). On hardware and the UNIX environment of the INGRES project, see Stonebraker, "Retrospection."

59. Stonebraker, "Retrospection," 232.

60. Ibid., 228.

61. Ibid.; Michael Stonebraker and Gerald Held, "B-Trees Re-examined," *Communications of the ACM* 21 (1978): 139–43. The INGRES group later rethought the decision in favor of B-trees. See Michael Stonebraker and Lawrence A. Rowe, "The Commercial INGRES Epilogue," in *The INGRES Papers: Anatomy of a Relational Database System*, ed. Michael Stonebraker (Boston: Addison-Wesley, 1986), 63–82, 76.

62. This also shows the unexpectedly large number of papers submitted to the International Conference on Very Large Data Bases, which took place in September 1975 in Framingham, Massachusetts. See *Proceedings of the 1st International Conference.*

63. Which meant, not least, that Codd had to continually publish new versions of his "original" concept. See Gerald Held in the foreword to Stonebraker, *The INGRES Papers.* Moreover, Codd seems to have been little involved in the developmental work on System R. The IBM developers' habit of referring to Codd as the founding father of the community must also be related to his being kept at arm's length from the project.

64. Stonebraker, "Retrospection," 230.

65. K. P. Eswaran and D. D. Chamberlin, "Functional Specifications of a Subsystem for Database Integrity," in *Proceedings of the Conference.* "To form a good basis for integrity control, a data sublanguage should be high level,

non-procedural, relationally complete, and should allow for the description and manipulation of sets of tuples, or records, in a single statement" (53). See also E. Allman et al., "Embedding a Relational Data Sublanguage in a General Purpose Programming Language," *ACM SIGMOD Record—Proceedings of the Conference on Data: Abstraction, Definition and Structure* 8 (1976): "A high level, non-procedural language, QUEL, for database query and update has been implemented in INGRES" (25).

66. Stonebraker et al., "Design and Implementation," 191.

67. Astrahan and Chamberlin, "Implementation," 582.

68. Arnold E. Keller, "The Man behind Systems at Shell Oil," *Business Automation* 7 (1962): 20–24, cited in Thomas Haigh, "'A Veritable Bucket of Facts': Origins of the Data Base Management System," *SIGMOD Record* 35 (2006): 34.

69. Alfred Eisenpreis, "Beispiele der Zusammenarbeit von Theorie und Praxis in einem Handelsunternehmen," in *Wissenschaft und Handel: Der Brückenschlag zwischen Theorie und Praxis, 4.–7. Juli 1966 in Rüschlikon-Zürich*, ed. William Applebaum (Bern, 1967), 133, my translation.

70. For a discussion of the role computers played for the transformation of banking in the last third of the twentieth century, see David Gugerli, "Data Banking: Computing and Flexibility in Swiss Banks 1960–90," in *Financial Markets and Organizational Technologies*, ed. Alexandros-Andreas Kyrtsis (Basingstoke: Palgrave Macmillan, 2010), 117–36.

71. Roland Barthes, *S/Z: An Essay*, trans. Richard Miller (Paris: Éditions de Seuil, 1974), 5.

72. "I think that a narrator, as well as a poet, should never provide interpretations of his own work. A text is a machine conceived for eliciting interpretations" (Umberto Eco, "An Author and His Interpreters," in *Reading Eco: An Anthology*, ed. Rocco Capozzi [Bloomington: Indiana University Press, 1997], 59).

73. Barthes, *S/Z*, 5–6.

74. Umberto Eco, *The Open Work*, trans. Anna Cancogni (Cambridge, MA: Harvard University Press, 1989); Susan Sontag, "Against Interpretation," in *Against Interpretation and Other Essays*, ed. Susan Sontag (New York: Picador, 1996), 3–14; Hans-Georg Gadamer, *Wahrheit und Methode: Grundzüge einer philosophischen Hermeneutik* (Tübingen, 1975).

75. Michel Foucault, "What Is an Author?," in *Textual Strategies: Perspectives in Post-Structuralist Criticism*, ed. Josué V. Harari (Ithaca, NY: Cornell University Press, 1979), 148; Roland Barthes, "The Death of the Author," *Aspen*, nos. 5–6 (1967); and Wolfgang Iser, *Die Appellstruktur der Texte: Unbestimmtheit als Wirkungsbedingung literarischer Prosa* (Konstanz, 1970). See also Harald Weinrich, "Für eine Literaturgeschichte des Lesers," in *Literatur für Leser*, ed. Harald Weinrich (Stuttgart, 1971), 23–34.

76. Dieter E. Sattler, ed., *Friedrich Hölderlin: Sämtliche Werke. Frankfurter Ausgabe*, Historisch-kritische Ausgabe (Frankfurt am Main, 1975), 16–20.

77. http://www.hoelderlin.de/materialien/html/marginalien.html.

78. Eco, *The Open Work*, 90.

79. Gil Grissom in "Toe Tags," *CSI*, May 10, 2006, season 7, episode 3.