Relational and Non-Relational Models in the Entextualization of Bureaucracy : Computational Culture

Abstract

To what can we ascribe the commercial dominance and organizational ubiquity of relational databases in the 1990s and early 2000s? By integrating perspectives from classical sociological theory, the history of administrative computing, and linguistic anthropology with papers and proceedings from the nascent database research community, this paper argues that the success of relational databases (and their corresponding tabular data models) can be ascribed to three distinctive components: 1) a *semiotic* distinctiveness from previously-existing (and now largely forgotten) hierarchical and network models; 2) a *bureaucratic* correspondence with practices of file organization and index-based retrieval; and 3) support for *transaction processing*, which facilitated adoption by a wide variety of organizations in commerce, banking, and finance. These three aspects are in turn used to suggest ways for understanding the potential significance (or lack thereof) of an emerging 21st-century market for non-relational and/or "Big Data" database systems.

Introduction

This paper addresses data management practices of the late 20th and early 21st centuries, specifically in the context of their use by large formal organizations. The most significant processual development in this period is the rise and subsequent dominance of data management systems using the relational model. Such relational database systems—which organize information conceptually in a similar manner to the tabular layout of a spreadsheet, but with concurrent access, transactional reliability, and a flexible querying interface—ultimately became the dominant technology for storage and retrieval of structured data in commercial businesses and the de facto standard on the web. They are also firmly at the technical core of a vast, global transformation of enterprise data processing and management that has taken place in recent years.

The relational model was far from an overnight success. Between its introduction to the nascent database research community in 1970 and the vibrant competition of functional implementations in the 1980s and 1990s lies a period of great argument and discussion over what amounts to a question worthy of the highest philosophy: what is the most appropriate representation of entities and their relationships in the world? This intellectual debate differed from others in that the answer was primarily determined by the interests of large multidivisional businesses, whose organizational difficulties a variety of computing firms, led by IBM, had long been dedicated to resolve. The goal of this paper is to connect bureaucratic practices (as initially theorized by Max Weber) to the distinctive properties of the relational database model, which are contrasted to previous (hierarchical and/or network) forms of data representation and management; and in turn to use this connection to suggest the possible implications of the 21st-century return of non-relational data models, for both enterprise organizations and the study of said organizations (or other structures) by social scientists.

Relevant Aspects of Weberian Bureaucracy

A cornerstone of classical sociological theory, Max Weber's comprehensive generalization of bureaucracy in his *Economy and Society* has influenced generations of scholars in the study of this most organizationally complex and technologically intricate of social structures. In his emphasis on the fundamental role of efficiency (in stark contrast to the already-predominant pejorative characterization of 'bureaucracy' as inefficient), Weber wove together a variety of seemingly-unrelated practices as aspects of a singular strategy which he called *rational-legal domination*. While the subsequent hundred years have brought a variety of confrontations and disputations of the relevance of Weber's "ideal type" of bureaucracy, from empirically-based "tests" to more contemporary theoretical revisions, few have attempted to supplant more than a small

subset of his typology at a time. But in our case, we shall only be concerned with the aspects of Weber's theory of bureaucracy which can be fruitfully applied to the historical development of electronic data processing and database management systems.

In an effort to be simultaneously general and specific, Weber addresses multiple overlapping aspects of bureaucracy without ever giving primacy to one or another: it is a condition simultaneously (but not preferentially) characterized by jurisdiction, exhaustive rules, hierarchical authority and management, and the maintenance and control of 'the files'. For our purposes the truly radical element of Weber's conception of bureaucracy is this singular emphasis on the importance of written documents, for it is in the development of database management systems that the use of "written documents"—today "written", of course, among and along an unfathomable variety of technological strata, including (at the lowest levels) disk drives, memory stores, and computer networks—achieves an organizational coextensivity unimaginable in Weber's lifetime. But even at the time, Weber saw the 'office' (*Bureau*)—"the combination of written documents and a continuous operation by officials"—as "the central focus of all types of modern organized action".

In the course of his argument, Weber discusses at length the bureaucracy's fixed-salary, full-time official positions; notes the necessary presumption of a monetary economy; observes the dramatically increased scale of administrative tasks (in part via increased means of communication); and also considers at length what he calls the "technical advantages" of bureaucratic organization, which will be especially important for us. These technical advantages include:

Precision, speed, unambiguity, knowledge of the files, continuity, discretion, unity, strict subordination, reduction of friction and of material and personal costs—these are raised to the optimum point in the strictly bureaucratic administration, and especially in its monocratic form.

It should be apparent to anyone familiar with the centrality and feature set of database technologies for all large organizations in the late 20th and early 21st century that we are dealing here with the first legitimate theorist of "data society"; for Weber's list of aspects of the highly rationalized organization can also be plausibly read as an promotional list of features of a present-day commercial database.

A further relevant topic regarding bureaucracy is its remarkable indestructability: "Once it is fully established," Weber says, "bureaucracy is among those social structures which are the hardest to destroy." For him this is a direct result of the apparatus of authority combined with the functional specialization necessary at every position; he does not explicitly refer to the concentrated *bureau* of files as contributing to the formal organization's improbably longevity. We can see further than he, however, and note the advances in data replication and storage which (with sufficiently continuous technical maintenance) gives the files an even greater (and more valuable) stability.

Therefore, we can conclude by enumerating the aspects of bureaucracy which are likely to be most relevant for understanding the both early electronic data processing practices and also those of the database management systems which followed them in the latter half of the 20th century:

1. The unification and centralization of records.

2. The efficient querying and processing of records.

To this we will add a third aspect which for Weber is not a generic property of all bureaucracy, but only for those organizations involved in what he called "rational economic profit-making"—the practice of *capital accounting*:

3. The reliable and secure tracking of internal and external economic transactions.

Keeping these three particular volitional goals for the enterprise in mind, we can address the corresponding

benefits or drawbacks of relational and non-relational databases as they emerged, dominated or receded—and in some cases, returned—beginning with the stirrings of high-profile debate in the (primarily Northern Californian) database research community of the 1970s. But as with any technology or set of technologies, the sociotechnical landscape of database management (much less *data* management, which would include the storage of less explicitly structured content like email) is not one of continuous progress but of a dynamic heterogeneity. While managerial literature past and present may offer the illusory vision of a business world progressively converging on the "best practices" of novel technologies, the reality is that—as seen in the Y2K scare—some systems, developed at great cost in the mainframe era, continue to serve their purpose today, on updated (or simulated) operating systems and hardware (though many have undoubtedly been supplanted by enterprise resource planning systems). The result is that a single database management system rarely completely prevails in any large organization; and many contemporaneous smaller firms continue to make do with older functioning systems; or—as in the case of many 21st-century startups—adopt more recently-developed database technologies at the outset. As such, one might consider this essay a contribution to the social history of "popular" database culture, which considers the relational systems which generated the economic success of, e.g., Oracle and SAP at the expense of many smaller software companies and a plethora of less-widespread information retrieval applications.

The Role of Computing for Bureaucracy

Following the lead of James Cortada, we should consider the history of computing not as the history of one monolithic tool considered independently from its various manifestations and the uses to which they were put, but as a history of a heterogeneity of *applications* (by which we mean the social use of various tools and techniques—or the interactive embodiment of such practices as 'software'—for specific goals). We can then distinguish between the developments in the broader history of computing wrought specifically by scientific and engineering applications—including the first electronic, stored-program computers—and those changes driven by business and government applications with very different organizational goals, which began with simple efficiency of data processing but came to include the need for higher-level programming languages, larger data storage, and resistance to operational failure. While the scientific precursors have, within the history of computing, traditionally been given precedence of importance over business-centric perspectives, recent works have emphasized the sensible idea that the computer industry (led over a long period of time by IBM) should be seen primarily as in direct continuity with the office equipment industry (led over a long period of time by IBM). While a strict logical separation of scientific/engineering applications and business/government applications is not fully warranted—innovations on either side often cross-fertilized, especially during wartime and within the research departments of IBM which, by the 1960s, was directly catering to both markets with the same products—the quantities of data used by managers and administrators traditionally outstripped those of scientific datasets.. Lars Heide comments that for punched-card applications in the 1930s, a large scientific calculation might have used "several hundred thousand punched cards.. in contrast to the several tens of millions of cards consumed four times a year by the Social Security Administration" (Heide, *Punched-Card Systems and the Early Information Explosion, 1880-1945* (Johns Hopkins University Press, 2009), 250-251). For a dual timeline of developments in scientific and commercial computing from the 1950s onward, see R.L. Nolan, "Information Technology Management Since 1960," in *A Nation Transformed by Information: How Information Has Shaped the United States from Colonial Times to the Present,* edited by A. Chandler and J. Cortada (Oxford University Press, 2000).]

It is therefore of great interest that much of the early technological change within commercial and government bureaucracies—the widespread adoption of keypunches, electric tabulators, printers and sorters in the punched-card lineage of Herman Hollerith's innovation —occurred contemporaneously with an absolutely radical transformation of commercial organization, a story vividly told in the historian Alfred Chandler's influential book, *The Visible Hand*. Chandler's history explains the rise of the large multidivisional business

enterprise—which, by 1950, had become the standard form of what he calls "managerial capitalism"—from its origins in the small, traditional, family-owned firms that defined commerce at the beginning of the 19th century. He does not explicitly focus on developments in data processing and management; he is, instead, explaining the very foundations and processual creation of what we now call 'data processing' and 'management'. He describes how in 1840 "the most advanced accounting methods.. were still those of [14th-century] Italian double-entry bookkeeping", and that "nowhere was the press of business enough to cause a merchant to delegate any of his tasks." But such was the rise in complexity, communication, and safety concerns that came with industrial development—and railroad transport especially—that in the following decades, new levels of delegation and control became necessary. Chandler describes one pioneering railroad superintendent who in 1853 not only initiates impressively Weber-esque principles of administration and subordination, but also puts into place a system of regular hourly and daily reports from engineers and rail agents via telegraph. (Notice here that a process of physical decentralization appears to simultaneously demand a greater logical centralization of knowledge, management and control.)

Two computing-minded business historians subsequently influenced by Chandler's suggestive analysis are JoAnne Yates, whose monographs examined the transformations in internal communication technologies and techniques through 1920 via case studies of the Illinois Central Railroad, Scovill, and Du Pont, as well as the insurance industry; and the former IBM researcher James Cortada, with his books on the data-processing technology of the early office appliance industry as well as a three-volume series on the adoption and use of computing technologies in various organizations and commercial industries Thomas Haigh has also published an important series of articles that also adopts the data-processing perspective, and (most relevantly) builds up to a history of database systems.The historian Michael Mahoney describes these developments in business-oriented computing history as helping to lead away from a history of the computer and toward a history of software

We can thus see evidence, in the last two decades, for a trend against a historiographic bias for 'electronic computing' and a corresponding trend towards 'data management and processing'—and my argument would be that favoring the former lineage necessarily leads to an overemphasis on the centrality of the Von Neumann stored-program architectures (and the related formal model by Turing), and a corresponding underemphasis on the Hollerith/IBM lineage of record-oriented data processing for commercial businesses which preceded and accompanied those developments. For just as Shannon's formalization of 'information' was unconsciously adopted by generations of misguided popular appropriators, we can see that Turing's formalization of 'computing' as the dynamic execution of machine instructions—symbolically encoded alongside non-executable data on a hypothetical infinite tape—itself leads to a notion of computers which emphasizes (the theoretically formalizable) programming and algorithms over the management and procedural processing of data. Neither side represents the 'essence' of computation because there is none—computation (like all technologies) has always been a heterogenous mix of tools, techniques, social formations, and volitions, and one can claim to find its origins in the astronomical tables which inspired Charles Babbage, or the statistical tables which inspired Herman Hollerith. What seems significant here is not what computing "really is", but instead the fact that both paths lead back to *tables*.

The Relational Model in Context

For the managers of large U.S. firms, the decade of the 1960s could be characterized as a time of great, and unfulfilled, technical promise. As Thomas Haigh describes, a powerful mythology among managers of the day was the idea of a *Management Information System* (MIS) which would centralize all information then being acquired and processed independently in separate divisions of the organization; this dream was largely not achievable using the technology and techniques of the era, although some complex proprietary data management systems had begun to be developed within individual large businesses. Examples include IBM's

own Information Management System (IMS), developed with North American Aviation (with initial representatives from Caterpillar Tractor) to handle the hundreds of thousands of parts in the bill of material for the Apollo program's spacecraft; and the Integrated Data Store (IDS), developed by Charles Bachman for General Electric's myriad manufacturing operations and first used in 1964. The former, IMS (still in use today, for reasons we will explain later), became known as the exemplar of the hierarchical model of database systems, due to its required representation of entities in a tree-like manner branching from a root. The latter, Bachman's IDS, was the starting point for what became known as the network model, named as such because entities could be related in graph-like directed configurations, and not just hierarchically.

It is not insignificant that these early database systems were predicated on large-scale manufacturing operations; the interest in databases was originally a symptom of only the most complex organizations, as they struggled from the weight of the subset of reality they sought to control, and searched for a stable ground (or "base") upon which it could support interrelated activities. Even before the possiblity of integrating data from multiple departments (operations, accounts payable/receivable, payroll, billing, etc.)—and certainly before the concept of integrating across multiple geographic divisions—single departments had already reached the limits of existing systems.

While this close connection between large commercial manufacturing firms and the development of database technology has presumably been long apparent to specialists, it bears mentioning just how closely related the concerns were. (Obviously, companies like IBM, RCA and Honeywell were also manufacturing computers and were thus doubly involved.) The main computing consortium of the 1960s, Codasyl (Conference on Data Systems Languages), had an Executive Committee predominantly populated by represented by commercial businesses as opposed to academic institutions, with representatives from all the main computer firms as wel as representatives from metals (U.S. Steel), chemicals (DuPont, B.F. Goodrich), insurance (Prudential Life), and transportation (General Motors). Codasyl's Programming Language Committee had been responsible for the formulation (in 1959-1960) of the widely used Cobol, a "Common Business-Oriented Language" whose syntax was intended to be more readable to managers than scientific programming languages like Fortran. It was their Data Base Task Group (DBTG) subcommittee—founded and initially chaired by W.G. Simmons of U.S. Steel—that was tasked with designing a specification for general database access; the DBTG's specifications in 1969 and 1971 were influenced strongly by both Cobol and committee member Bachman's IDS. However, the Codasyl DBTG specification was not universally acclaimed, especially by those who noted its complexity and, especially, its insufficient separation of programmer-level operations and the structuring of data on the physical device.

It is in this context of this building of commercial demand—and correspondingly imperfect data management solutions (as well as a formal specification met with a not-insignificant amount of skepticism)—that Edgar F. "Ted" Codd's influential article, "A Relational Model of Data for Large Shared Data Banks", was published in the Association for Computing Machinery's flagship journal *Communications of the ACM*. Although the idea of the relational model took some time to catch on, as the decade progressed Codd's proposal came to occupy a prominent role at the controversial forefront of database research, despite not yet being implemented, commercially or otherwise. Codd, employed at IBM's San Jose Research Laboratory, had previously worked both inside and outside IBM on a variety of projects ranging from electronic calculation, data processing, and multiprogramming (the ability to run multiple jobs simultaneously on a single computer).

But unlike many IBM researchers who had been trained primarily in engineering, Codd had studied mathematics and chemistry at Oxford, and had recently completed a Ph.D. in Communication Sciences at the University of Michigan, working under Arthur W. Burks on cellular automata. Burks was a professor of philosophy who had previously worked at the University of Pennsylvania on the ENIAC, the world's first general-purpose computer; rather distinctively, Burks helped found Michigan's computer science program in

the mid-1950s—then the 'Logic of Computers' program—while simultaneously editing the collected papers of Charles S. Peirce. It would seem that Codd's immersion in the intricacies of both computer engineering and symbolic logic allowed him to view the pragmatic idea of data modeling in a distinctive way; but the resulting formal style in his 1970 paper also slowed uptake among less-mathematically-inclined researchers. The story of database research in the subsequent decade of the 1970s can be summarized in the gradual diffusion of the idea of the relational model and its (eventual) implementations; and eventually, the relational model made aspects of the long-fantasized "total information system"—extended to its contemporary form as the wholly centralized Enterprise Resource Planning (ERP) installation—possible. However, one can also be more detailed about the source of its transformative power.

I will argue that the success of the relational model can be understood on three more-or-less independent axes, which I will label *semiotic, bureaucratic, and transactional*:

1. The relational model differs primarily in its wholly symbolic and tabular representation to the user, as opposed to the explicitly encoded referential relations of the hierarchical and network models. This fundamental semiotic difference produces a highly valued effect recognized more typically as 'data independence'.

2. Somewhat surprisingly, the relational model mapped onto the cognitive practices of traditional administrative batch processing better than the IMS-style hierarchical model or the DBTG-style network model (which were both, in part, explicitly trying to preserve the batch processing "record-at-a-time" logic).

3. The relational model, because of its symbolic representation of entity relations, was also highly amenable to the formalization of concurrent transactions, a technology which emerged and improved contemporaneously in the late 70s and early 80s. The ability to process atomic transactions at high speed while maintaining a consistent state was an enormous selling point for database management systems regardless of whether or not they used a relational model.

1. The Semiotic Aspect of the Relational Model

Codd's 1970 paper begins with the statement:

Future users of large data banks must be protected from having to know how the data is organized in the machine (the internal representation).

In this statement Codd is addressing the call for a greater separation between what was at the time called *the logical* and *the physical*. The *logical* corresponded to the programmer (or "user")-level perspective of the data; the *physical* corresponded to some metaphorically 'lower' level of internal hardware (for example, specific locations on a disk drive). Before IBM's unbundling and the emergence of "software" as a commercial industry distinct from computer manufacture, the boundaries between one and the other could be fairly loose, and certainly varied impressively from computer to computer and system to system in the 1960s. But the history of revolutionary computer applications is, in large part, a history of techniques which successfully compartmentalize symbolic layers: such methods today (in software engineering) variously go under the name of *abstraction*, of *encapsulation*, of *protection*. In 1970 Codd spoke of "data independence" (i.e., keeping the representation of the data separate from the underlying structure on disk), and in the 1960s a notable issue was of "machine independence" (developing applications which could function not just on IBM's computers, but on those of other manufacturers). For example, the "high-level language" of Cobol allows a programmer to modify records without specifically acknowledging the underlying operating system or hardware. Slightly predating such terms (but again based on this notion of independence between artifactual layers) was the notion of "generalization", the idea that one could develop methods for processing data—sorting, for example—which did not depend explicitly on the particular deck of punched cards one was dealing with.

Codd's relational model was very distinctive from then-existing database models in having only one formal user-level conceptual data type: the *relation*, which can be thought of as a simple table (where the ordering of rows is unimportant). This allowed Codd to adopt certain algebraic operators as a starting point for a hypothetical high-level, interactive data-retrieval language. So, for example, the algebraic *projection* operator (of dimensions in a relation) became analogous to selection (of columns in a table); and the *natural join* operator could be used to connect tables into new tables, based on the shared values or one or more columns. Codd noticed that the use of this so-called relational algebra provided a potentially vast simplification over the often-complex programmatic traversal of data structures necessary to retrieve data (at the user level) in existing hierarchical- and network-based systems; and in contemporaneous and subsequent papers, Codd developed the idea of using these operators to interface with a proposed 'casual user' of databases. (This hypothetical query language would ultimately be incompletely realized in the now-ubiquitous SQL.)

But how did Codd's proposal of representing both entities and relations as tables help with the data independence problem? In order to understand this, we can fast-forward to the ACM SIGFIDET conference of May 1-3, 1974 in Ann Arbor, where the differences between the relational model and the network model from the DBTG network model were most explicitly presented, in the form of a gentleman's duel between Codd and Charles Bachman. The two had publically clashed in the previous year's SHARE conference in Montreal, where Bachman presented a draft version of his ACM Turing Award lecture. Entitled "The Programmer as Navigator", Bachman's idealized interlocutor for database interaction is not at all the 'casual user' that Codd envisioned, but of an advanced programmer who "picks his way through the data to resolve an inquiry or to complete an update." At the end of the talk, Codd rushed to the microphone and, while being relatively civil, admitted to "disagreeing entirely" with Bachman's vision. Instead, Codd said—referring to his own relational model—"we now have an opportunity to reverse the trend of making the logical view more and more complicated".

For the 1974 Ann Arbor conference, both Codd and Bachman would be giving presentations, alongside papers by supporters of their respective approaches. Codd worked with Chris Date (an early, and subsequently life-long, disciple of the relational model from IBM Hursley) to draft two papers: one—primarily by Codd—describing the relational approach's potentially superior support for non-programmers and the other (primarily by Date) describing its benefits for programmers. Codd's paper emphasized the relational model's goals as: 1) simplifying the logical data structures (i.e. using only tables as opposed to the cornucopia of data types in the network model), 2) allowing both programmers and non-programmers "to store and retrieve target data *without having to "navigate" to the target*" (emphasis in the original); 3) to introduce an English-language interface for truly casual users; and 4) to provide rules about authorized access (i.e. which user can see what data) and integrity (e.g. keeping the data in an appropriately consistent state) which can be specified separately from the logical view. While Codd was somewhat misguided about the near-term possibility of implementing a responsive natural-language interface (today, those working in business intelligence still need to learn either SQL or a clunky interface which can generate it), each of his points was a direct attack on a problematic aspect of the network model with respect to data independence; Bachman's side simply did not have a comparable vision for the use of the network model by "casual", non-programming users.

Date's paper, on the other hand, used a novel kind of diagrammatic rhetoric to contrast the two approaches, as seen in Figure 1. By comparing the simplicity of the logical views of the relational model to the complexity of the network model, Date vividly demonstrates significant differences, which Bachman would often try to play down by suggesting that the two models could somehow be reconciled. Indeed, I would argue that the diagrams in Fig. 1 can provide an immediate understanding, even to the non-specialist, of what makes the two approaches fundamentally different. As can be seen in the top half of Fig. 1, the relational model does not have, or explicitly require, any referential relations between the part (represented by table *p*) and the supplier

(represented by table *s*). Instead, their relationship is represented by another table (labeled *sp*). With this third table, the association between attributes of *s* and *p* could be generated dynamically, as needed by the user, as opposed to existing as pointers at the logical level or (as in the computer programming sense of "pointer) at the physical level.

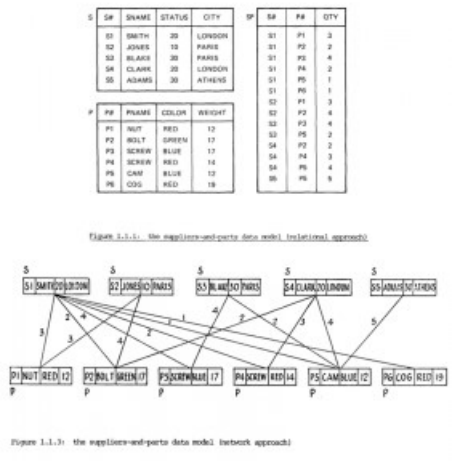

FIGURE 1: Date and Codd's diagrammatic comparison of the logical views of a relational database (top) and of a network database (bottom). From Date and Codd, "The Relational and Network Approaches: Comparison of the Application Programming Interfaces", 89-91.

One can compare this situation to the diagram of the network model (the bottom half of Figure 1). Here, it is made clear that a network-based system of absolutely minimal complexity looks—on paper anyway—like something of a mess. Here, every supplier appears to have a direct, referential connection to each of the parts it supplies, and those referential connections themselves need to store extra information (in this case the number of parts supplied). But in fact, the situation is even worse than this. Date also provided diagrams showing the user's view of a DBTG implementation of the network model, as shown in Figure 2. On the left side of Figure 2, we see that in order to be able to relate parts to suppliers in both directions in a DBTG system, we need to explicitly create two objects to connect the records, *S-SP* and *P-SP* (called 'owner-coupled sets' in the DBTG specification). On the right side of Figure , we see not the implementation of this model, but the actual programmer-level logical perspective (although one can imagine the physical level looking nearly identical—this was one of Codd's complaints with respect to the network model's lack of data independence). Date does not even have room to show all the links necessary for just these few records. Imagine adding a third entity to the table—e.g., including the warehouses in which the part is stored; an uncomfortably intricate spaghetti emerges. This complexity is what needs to be 'navigated' by the programmer, to use Bachman's term. By comparison, imagine adding a third entity to the relational model: one would add a table *W* listing each warehouse's attributes, and a table *WP* which related warehouses to the parts they stored, producing five tables in total—a comparative simplicity, and still readable at the logical level.

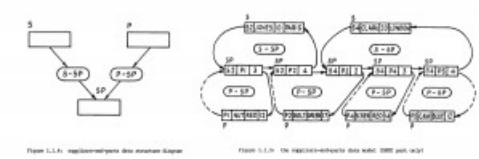

FIGURE 2: Date's depiction of the Codasyl DBTG network model. From Date and Codd, "The Relational and Network Approaches: Comparison of the Application Programming Interfaces", 92-93

Moreover, it should be clear that while in both model's representations, the parts and suppliers have as an attribute a unique, identifying symbol (*S1, S2, P1, P2*, etc.), it is only the relational model which takes explicit

advantage of this identifier to represent relationships. A fundamental advantage of the relational model can be located, and this is it: the *referential* relationship of part and supplier is instead represented using *symbols*. The distinction between the two models is thus fundamentally *semiotic*; where the network model's pointers mimic the indexical real-world physical relationship between part and supplier, the relational model represents that relationship in an explicitly symbolic, tabular form. This tabular representation is by no means "natural", but it might be said that it is "natural" within a certain type of social structure—such as a commercial or administrative bureaucracy—which (as we know from the business historians) had been working and thinking with records and tables for many decades. I would argue that this focal difference—of transducing reference to symbol—is at the heart of "data independence" in all of its forms.

So the crucial difference between the network model and the relational model should be clear. Where the network model enforces referential (i.e., pointing) links between entities at the logical level, the relational model enforces the *absence* of such reference. In this way, we can say that what the relational model allows for is a sort of freedom in recontextualization of the (entextualized) database artifact. This freedom is realized in the so-called "expressiveness" of relational query languages like SQL, which (given an appropriately normalized design) allow one to relate — via joins and projections — new entities with every interaction. From a naïve perspective, this freedom would appear to be significantly less efficient (and such a complaint was taken quite seriously when computers were as comparatively slow as they were in the 1970s); it would seem that if I want to, for example, find the color of all of the parts supplied by supplier #2, I must exhaustively search the (unordered) part table P to locate each row based on the part identifier, as opposed to simply following an pointer/link. The answer is that such searches, while definitely slower than following a pointer value, can be quite efficient; the technology which allows it is relatively independent from Codd's contribution, as it was already well known, having been used from the early days of print. This technology is known as an *index*.

2. Bureaucractic Aspects of the Relational Model

It would be difficult (if not pointless) to attempt to undertake a history of the origin of the practice of assigning individual numbers to observed entities in the world, for this technique seems rather natural to the numerate and is thus likely to have been independently re-discovered in a variety of periods and settings within cultures with writing practices. However, there is reason to consider the (written) numbering of things as a fundamentally important technique, for it makes a variety of otherwise extremely difficult tasks possible. Consider, for one, the ability to quickly discover which pages of a book discuss a given subject; this is made possible by what we call an *index*, which maps subject terms to a list of page numbers. Such subject indexing is a human art, but it is dependent on page numbers as a prerequisite of its technique. The dependence on some kind of ordered identifier is, of course, commonplace in many organizations which depend on the accounting of *people* as opposed to pages. While smaller bureaucracies might simply use alphabetical ordering and resolve identity conflicts on a case-by-case basis (i.e. disambiguating two files for "John Smith"), for more complex systems like the national census, or for life insurance agencies, a unique integer identifier is consistently used.

In this section, I will describe how a particular data structure based on the use of unique identifiers—the *B-tree* —implements a similar kind of fast, indexed lookup, and thus provides a certain continuity of technique with the standard practices of the rationalized bureau. Indeed, for database management systems to succeed, 'the files' of a computer-assisted bureaucracy should be at least as accessible as those with rooms full of file cabinets. But before the innovation of random-access disk storage (which inspired the development of databases), what we now think of as computerized "data management" was in a primitive state. The use of magnetic tape for storage encouraged the rather limited use of that medium as a linear sequence of records, one after another, just as punched card records had been processed one after another—perhaps tabulated, perhaps updated, perhaps sorted and output to another tape—but always in serial batches. (See Fig. 3 for an

illustration, from a 1959 paper.) By comparison, the "random access" of disk storage—by which any given sector of the disk could be retrieved within a certain bounded window of time (as opposed to the lengthy playback necessary on tape-based media)—was crucial to the conception and implementation of applications capable of the storage and retrieval of multiple types of related records.

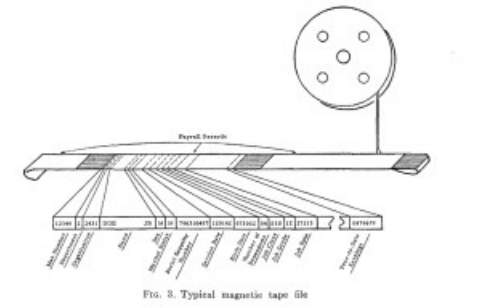

FIG. 3. Typical magnetic tape file

FIGURE 3: From McGee, "Generalization: Key to Successful Electronic Data Processing."

But the feature of having random access to some given sector of a disk does not inherently reduce the complexity of inserting, deleting, or even searching for information from a particular record (say, e.g., an employee with identification #12345). Without some mechanism for efficiently translating from the identification number to a disk location, one would (just as on a linear tape drive) start at the beginning of the disk and scroll forward until one found the given record. The eventually widespread solution to this problem involved the progressive realization of a data structure which, in its minor variations, would come to be known as "the ubiquitous B-tree" The B-tree is portrayed in Fig. 4: in order to find a record with a specific unique integer identifier (or, in relational model parlance, a primary key), we compare that integer with the values at each level and traverse down to the appropriate leaf node, which eventually leads to a data page with the actual record stored in it; note that the number of traversals is small as compared with the total number of potential records stored.

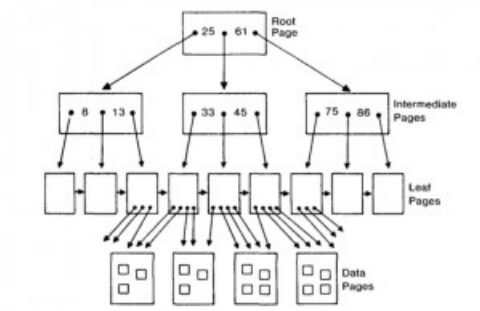

FIGURE 4: A B-tree index for a relation using an integer primary key, as used in the System R relational database. From Chamberlin et al. "A History and Evaluation of System R," Communications of the ACM 24, no. 10 (1981): 641.

The novelty of the B-tree's design — in comparison to previously existing tree-based data structures— is its guarantee that no matter where records are inserted or removed, the tree remains 'balanced'—i.e., the expected search time is always logarithmically proportional to the number of records. This is performed by an algorithmic "re-balancing" which occurs in the case that a new node needs to be added to an already-full leaf node (this process is shown in Fig. 5). The result is a consistent, relatively fast way to go from a primary key to the corresponding data; and the branching search technique is not all that dissimilar from the methods of the traditional bureau (narrowing down from rows of file cabinets, to a particular cabinet, to a particular file). If—as we described in the previous section—the relational model's distinguishing feature is the transduction of referential relationships to records of symbols, a B-tree is a fast way to go in the other direction. Without this technique, relational models would likely have remained as inefficient as their detractors often predicted.

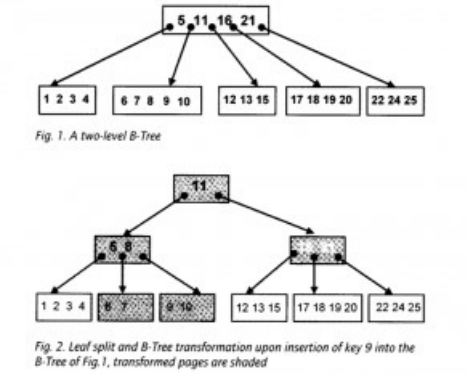

FIGURE 5: Diagram depicting the dynamic re-balancing of a B-tree upon inserting the value '9' into a full leaf. Both the leaf node and the root node split in succession, creating a new tree with three levels instead of two. From Rudolf Bayer, "B- Trees and Databases, Past and Future," in Software Pioneers, ed. M. Broy and E. Denert (Springer-Verlag, 2002), 235.

When combined with corroborating techniques like these efficient B-tree indexes, Codd's proposal of the relational model, in retrospect, seems ultimately less radical than (pragmatically) conservative. It is as if to say: computers are used most effectively for the processing of standardized records—that is, of structured and serialized sequences of symbols, extracted (i.e., entextualized) from their multivalent surrounding context. Therefore, make all data in the form of a standardized record. Date mentioned this rather explicitly in his 1974 SIGFIDET paper, describing the relational model's "closeness to traditional 'flat files'—i.e., to the enormous number of files which currently exist and are organized sequentially, especially on tape." Codd was even more specific during his interrogation of Bachman at the 1973 SHARE conference:

"In looking at the progress towards integration of files, we have noticed that the entities, previously processed separately, now have to be more and more heavily interrelated. This has resulted in the very elaborate data structure diagrams that we have seen displayed here.. Now it so happens that a flat file and a collection of flat files has all of the power you want to deal with the n-dimensional world. You do not need to introduce any separate concept of relationships: the pointer-style relationships that we see with arrows on [Bachman's] diagrams. It so happens that you can consider the entities like parts, suppliers, and so forth and relationships like 'so-and-so supplies such-and-such a part' in one way, a single way, namely the flat-file way. What are the advantages of doing this? By adopting a uniform flat-file approach you can simplify tremendously the set of operators needed to extract all the information that can possibly be extracted from a data base."

This simplicity of the relational approach—shrouded in appearance, for some, in the formalities of set-theoretic algebra and predicate logic—only became apparent to researchers at varying rates; its attempted implementations quickly converged on the need for indexing techniques like the B-tree which, indeed, possessed an isomorphism to existing bureaucratic practice for data retrieval. However, there was one further set of intellectual developments—occurring somewhat orthogonally from the central arguments about the relational model—which, by the end of the 1980s, finally allowed relational database systems to completely transcend their role as a promising object of theoretical interest, and instead to take a dominant position in the commercial software landscape, one for which it has largely yet to yield to newcomers. This was the formalization of the *transaction*.

3. The Transaction Abstraction in the Relational Model

"Capital is not a thing, but a social relation between persons which is mediated through things."

While Codd's 1970 paper title indeed makes reference to "Shared Data Banks", one should not be misled into believing that database management systems were born with the facility for concurrent, networked use. The technological landscape of multiuser systems over the 1970s was quite diverse, and close reading reveals that

many applications intending or claiming to support multiple simultaneous users did so at a coarse-grained scale which would seem unacceptable by present-day standards. For example, banks throughout the period came to rely more and more on computers for high-speed, overnight batch processing; but systems which could perform what became known as *transaction processing* (later *on-line transaction processing*, or OLTP) were only slowly beginning to emerge (as in, e.g., with relatively low-thoroughput ATMs.) The idea of concurrent interactions with computers was not new; the implementation of *multiprogramming*, a goal of computer manufacturers throughout the 1960s, allowed multiple users' jobs to run on the processor in an interleaved fashion as they alternately executed or waited for input or output from some resource to complete. But as larger random-access devices (combined with the aforementioned MIS mythology) inspired the development of database management systems, it was easy to envision such systems allowing for simultaneous reading and writing from more than one user. However, as we have seen, such work would need to be accompanied by a strong commercial demand.

As it turns out, the general problem of the limitations of written accounting of economic transactions had raised its head at many firms before the 20th century. The bound volumes of traditional double-entry bookkeeping were, as we would say today, single-user technologies. It is easy to imagine how the limitations on concurrency in traditional record-keeping might have been a bottleneck in the otherwise rapid expansion of the multidivisional organization. One early organized workaround was the *clearinghouse*, whose early manifestations (such as the Bankers' Clearing House in early 19th-century London) centralized end-of-day record processing among multiple institutions, settling accounts far more efficiently than were they to be processed independently. Ledgers were imperfect solutions to reliability and security, and themes of reliability (often under the label of "recovery" from inconsistencies or system error) and security became more prevalent in the database research literature in the late 1970s and early 1980s, as the debates on logical representation became—especially for the apostles of Codd at Berkeley's INGRES, IBM's System R group, and other relational database implementors such as the nascent Oracle—largely settled.

By 'the transaction abstraction' I refer to a set of concepts that coalesced into an increasingly formalized model in the 1970s and 1980s, and which has been differentially implemented on a variety of software systems ranging from the 1960s to the present. Some aspects of the basic concept of transaction, however, extend back to the development of contract law; this fact has been noted by both computer scientists and organizational sociologists. The transaction abstraction came, by the early 1980s, to be encompassed by a set of concepts using the acronym 'ACID':

- *atomicity*: Individual transactions either occur or—if aborted—have no effect.
- *consistency*: The data appears in a correct, valid state at all times.
- *isolation*: Transactions are isolated from one another; this is equivalent to the appearance of serialized execution.
- *durability*: Successful transactions persist and do not disappear or become corrupted in the case of failures.

These four concepts are not orthogonal (atomicity and isolation, for example, are closely related), and in fact it is worth noting that *consistency*, by virtue of having particular, sometimes complex, meanings for particular applications (such as making sure that debits and credits cancel out in an account transfer), is not a abstract, formalizable notion, but something whose semantics must be specifically ensured by the application's programmer. By comparison, the guarantees of atomicity, isolation and durability can be reasoned about more formally, under the guise of particular techniques such as *two-phase commit*, *serialization theory*, and *Undo-Redo protocols*.

It is a remarkable fact that many comparatively ancient computing systems which successfully implemented

transactional stability in the past have never completely gone away. From the airline booking system SABRE; to IBM's mainframe-based transaction monitor CICS; to IBM's hierarchical mainframe database IMS (whose logical design has gone thoroughly out of style, but which which provides newer facilities for interacting with relational data); all of these systems have their origins in the 1960s and—far from disappearing along with so many other ephemeral software technologies—continue to run today on modern mainframes, allowing large organizations to continue to run hundreds of thousands of lines of non-portable code. The notions of enforcing consistency and durability are crucial elements for any bureaucracy's shift away from paper-based bookkeeping, as it ensures that in many conceivable types of system failure, the data will still be successfully written (or successfully aborted). In the case of the relational model, transactions were often cited as a make-or-break feature for enterprise and/or financial usage of relational database products in the 1980s. In an interview, Don Haderle of IBM's System R project describes when System R's successor, DB2, came to be taken seriously:

"So when version 2, release 1 came out in 1988, there was an article in *Computerworld* that said "Relational is now ready for primetime." It was simply because we had the features that were there for doing transaction processing. The customers came back and said it's competitive for that environment now. Good enough."

Just as the intellectual debates regarding database models rotated around toy examples drawn from the context of large-scale manufacturing (Employee, Manager, Part, Supplier, Warehouse, etc.), the canonical example of the early transaction processing literature is the Debit-Credit transaction, which constituted a conceptual threshold for transactional sufficiency, while also reflecting the role of the banking industry, which held the earliest stake in the benefits of transaction processing. Fig. 6 shows an early diagram of the sort of staging that would characterize the transaction abstraction—here characterized by the ability to invertibly abort or "backout" of the process; and Fig. , from , shows how this "sphere of control" can be extended in theory to serial and parallel sequences of actions, using banking and manufacturing processes as examples.

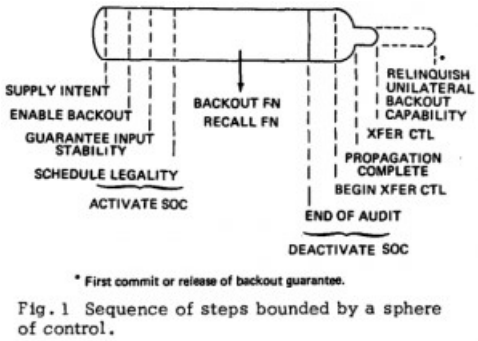

Fig. 1 Sequence of steps bounded by a sphere of control.

FIGURE 6: Charles T. Davies' early transaction abstraction, which he called a "sphere of control".

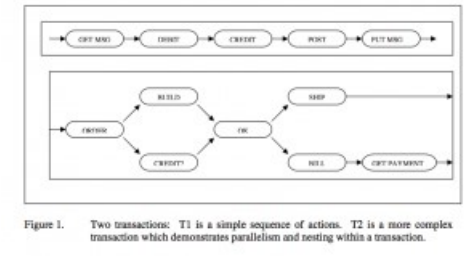
Figure 1. Two transactions: T1 is a simple sequence of actions. T2 is a more complex transaction which demonstrates parallelism and nesting within a transaction.

FIGURE 7: Jim Gray's transactions. Above, a simple debit-credit transaction. Below, a more complex manufacturing/billing transaction with parallelism. From Gray and Reuter, Transaction Processing: Concepts and Techniques.

Understanding the development of the transaction abstraction would give us great purchase into Donald MacKenzie's admonition to "open the black boxes" of global finance. But as we have seen with the debate on

relational vs. network models, it is not in fact always necessary to open up the black boxes completely—to the level of, e.g., the source code—for various competing individual implementations. One instead can derive great understanding of the transgressiveness of computational innovations (such as the relational model) in the absence of (now decades-old source code), through a historical examination of published (and unpublished) research and debate. However, we are at a serious disadvantage in that the techniques of transaction processing, unlike those of database management systems, have yet to be recognized by historians of computing and other social scientists as a locus of contemporary relevance. By contrast, what has been taken as an object of interest by social scientists are certain changes in economic phenomena, such as the vast increase in securities and derivatives exchange known as *financialization* and/or *marketization*; or the expansion of multidivisional enterprises to (seemingly decentralized) transnational operations variously concentrated on (intriguingly centralized) metropolitan financial centers (globalization). Such developments are often illustrated with graphs showing an exponential rise in, e.g., historical trading volume (see the left graph of Fig. 8), while seeking to describe such rapid change as a political or otherwise human-centric social process. It is thus little recognized that efficient and durable transaction processing—which permits the concurrent, orderly, reliable, and centralized serialization of economic exchange—is in fact a firm technical prerequisite of such financialization. It must be understood that the simple increase in speed of single computer CPUs is not sufficient to explain this orders-of-magnitude scaling of computer-centered economic exchange that has occurred since the 1970s. The sociological and historical theorization of the modern formal transaction, while necessary, has yet to occur.

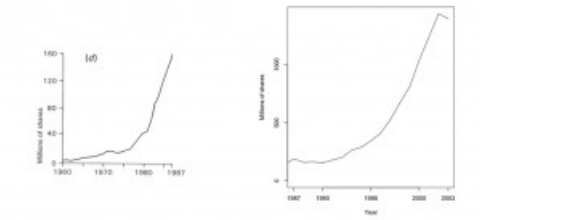

FIGURE 8: At left, New York Stock Exchange trading volume, 1960-1987, from David Harvey, The Condition of Postmodernity (Wiley-Blackwell, 1991). At right, NYSE average daily trading volume, 1987-2003 (Source: NYSE)

Conclusions and Implications: On Big Data

I have shown how the relational model for databases provides features appropriate to its origin in commercial manufacturing interests and other large organizations, and that for various reasons—a combination of a semiotic 'data independence' and the implementation of indexes and transactions—the relational database can be seen as an ideal medium for the entextualization of many forms of late-20th-century bureaucracy. To conclude, I will suggest that this understanding can potentially be lifted from its historical setting to help understand the relevance and potential directions of more contemporary debates regarding database practices. One such debate surrounds the significance, in the last several years, of the development of new kinds of non-relational databases, primarily to handle so-called "Big Data" (i.e., quantities of data which theoretically exceed the facilities of existing, centralized relational stores), many of which are quite different in design from the database models mentioned in this paper.

The sociotechnical phenomena presently going under the terminological umbrella of "Big Data" are, on initial inspection, reminiscent of the heterogenous complexities of the early days of commercialized database software, with a wide variety of contenders—some developed for and within established computing businesses with extreme data management needs, some from small startups—filling a complex, overdetermined online discourse with a mixture of genuine success stories and unverifiable claims. While the emergence of the first database management systems were closely correlated with the availability of random-access disk technology

which could support their physical structures, these new large-scale databases have correspondingly emerged alongside the inexpensive availability of large arrays of disks; of cloud-computing techniques which can dynamically provide or retract virtual storage on demand; and larger stores of main memory. But in general, these new non-relational databases tend to either relax the requirements of transactional stability (by, e.g., providing potentially less consistent results from a more decentralized database) or the relational requirement of purely tabular modeling (by, e.g. storing data as simple flat files of values, or as collections of key-value-oriented 'documents'.)

By considering data stores as mediating text-artifacts, we can say that whatever "Big Data" is, it clearly involves an order-of-magnitude increase in entextualization. But manufacturing processes, for example — despite their increased coalition via mergers into the late 20th century — have, for the most part, *not* undergone similarly-scaled order-of-magnitude increases in, e.g. parts, suppliers, products, or customers (although the involved firms may have been involved in mergers which were in turn facilitated by commonalities in database systems). So *what* exactly is being entextualized? The answer is that this data often represents—and this differs quite radically from the uses of database management systems in the past— the entextualization of events and entities not just "inside" the organization, but those "outside" as well. Previously, a customer ordering a product online would only be represented in the company database (i.e., as an entextualization) after their initial transaction, and be represented somewhat consistently thereafter —name, shipping address, item ordered, etc. However, other contextual aspects of their behavioral interactions with the organization can today be entextualized alongside the conventional transaction information—whether it be a list of the sequence of web pages the customer clicked on before they purchased the item; or other data associated with the user's cookie, as dynamically acquired and associated with data acquired via marketing firms; or even the weather in the user's area at time of purchase. In the case of contemporary social media sites, their fast-growing content is, in part, volunteered by users at scales improbable even to the transaction systems of late-20th century commercial businesses. The variety of potential entextualization from large decentralized data sources like the Internet is limitless, and, for those with deep pockets, so is the size of the "eventually consistent" distributed databases one can now build around such entextualizations.

On the other hand, it is not clear that what traditional bureaucracy—as a paragon of organizational control —needs, or wants, such infinite quantities of entextualizations. If we return to our proposed aspects of bureaucracy relating to data processing, we can see that with some of the "Big Data" non-relational databases we have (1) potentially reduced the degree of unification and centralization of the data (because consistency is not the same thing as a distributed "eventual consistency"); (2) potentially increased the efficiency of processing very large quantities of data (for there is typically still a facility for some kind of indexing); and certainly (3) diminished the support for reliable and serialized representation of economic transactions. These observations should give us some skepticism as to any recent announcements of the impending demise of relational technology (unless these drawbacks begin to be resolved legitimately by non-relational database systems).

And what can we learn from the "great debate" which posed Codd's relational model against Bachman's network model (as proposed by the Codasyl DBTG) in 1974? Only that the pragmatic merits of tabular, symbolic data storage would, for large organizations and a generation of enterprise software systems extending to the present—ultimately trump the explicit representation of referential relationships. This is, today, even true for the vast 'social network' of Facebook, which primarily relies upon a distributed relational database (a customized branch of the open-source MySQL), and not, for example, a network or graph-structured database. This tendency of both the traditional and the more novel enterprise to prefer the efficiencies and reliability of structured, transactional data representations over the network-graph data models currently so fashionable in

social-scientific literature could, I think, become a starting point for a new methodological self-analysis among digital humanists and social scientists. As we extend the explicit encoding of referential relations from single data types ('individuals' and their 'ties'; or 'papers' and their 'citations') to more elaborate sociotechnical scaffoldings, are we gaining analytical purchase from the added complexity of our datasets? Perhaps such extreme network-centric perspectives, in the future, will be considered a once-promising but ultimately misguided empirical fad—as in the not-dissimilar Codasyl DBTG proposal, of which no implementations currently survive.

Bibliography

Ashenhurst, Robert. "Report on a Meeting: A Great Debate." *Communications of the ACM* 17, no. 6 (June 1974).

Bachman, Charles W. "The Origin of the Integrated Data Store (IDS): The First Direct-Access DBMS." *IEEE Annals in the History of Computing* 31, no. 4 (2009): 42–54.

Bachman, Charles W. "The Programmer as Navigator." *Communications of the ACM* 16, no. 11 (November 1973): 653–658.

Backstrom, Lars, Paolo Boldi, Marco Rosa, Johan Ugander, and Sebastiano Vigna. "Four Degrees of Separation" (January 5, 2012). http://arxiv.org/abs/1111.4570.

Bátiz-Lazo, Bernardo. "Emergence and evolution of ATM networks in the UK, 1967–2000." Business History 51, no. 1 (January 2009): 1–27.

Bauman, Richard, and Charles L. Briggs. "Poetics and Performance as Critical Perspectives on Language and Social Life." *Annual Review of Anthropology* 19 (1990): 59–88.

Bayer, Rudolf. "B-Trees and Databases, Past and Future." In *Software Pioneers*, edited by M. Broy and E. Denert, 233–244. Springer-Verlag, 2002.

Bayer, Rudolf, and Edward M. McCreight. "Organization and Maintenance of Large Ordered Indices." *Acta Informatica* 1 (1972): 173–189.

Bergin, Thomas J., and Thomas Haigh. "The Commercialization of Database Management Systems, 1969-1983." *IEEE Annals in the History of Computing* (2009): 26–41.

Bernstein, Philip, Vassos Hadzilacos, and Nathan Goodman. *Concurrency Control and Recovery in Database Systems*. Addison Wesley, 1987.

Bijker, Wiebe E., Thomas P. Hughes and Trevor Pinch, eds. *The Social Construction of Technological Systems: New Directions in the Sociology and History of Technology*. MIT Press, 1987.

Braudel, Fernand. *The Wheels of Commerce: Civilization and Capitalism: 15th-18th Century*, Vol. 2. University of California Press, 1992.

Campbell-Kelly, Martin. "Victorian Data Processing." *Communications of the ACM*, no. 10 (October 2010): 19–21.

Çalıskan, Koray and Michel Callon. "Economization, part 2: a research programme for the study of markets." Economy and Society 39, no. 1 (2010), 1–32.

Chamberlin, Donald D. "Early History of SQL." *IEEE Annals of the History of Computing* 34, no. 4 (2012), 78-82.

Chamberlin, Donald D., Morton M. Astrahan, Mike W. Blasgen, Jim Gray, W. Frank King III, Bruce G. Lindsay, Raymond A. Lorie, James W. Mehl, Thomas G. Price, Gianfranco R. Putzolu, Patricia G. Selinger,

Mario Schkolnick, Donald R. Slutz, Irving L. Traiger, Bradford W. Wade and Robert A. Yost. "A History and Evaluation of System R." *Communications of the ACM* 24, no. 10 (1981): 632-646.

Chandler, Alfred D. *The Visible Hand : The Managerial Revolution in American Business*. Belknap Press, 1977.

Chang, Fay, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. "Bigtable: a Distributed Storage System for Structured Data." *ACM Transactions on Computer Systems* 26, no. 2 (June 2008), 4:1-4:26.

Codd, E.F. "A Relational Model of Data for Large Shared Data Banks." *Communications of the ACM* 13, no. 6 (1970): 377–387.

Codd, E.F. , "Relational Completeness of Data Base Sublanguages." In *Courant Computer Science Symposium 6 May 24-25 1971: Data Base Systems* (Prentice-Hall, 1972), 33–96

Codd, E. F. "Normalized Data Structure: A Brief Tutorial." In *Proceedings of 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control,* San Diego, California, November 11-12, 1971 (ACM, 1971), 1–17.

Codd, E. F., and C. J. Date. "Interactive Support for Non-Programmers: The Relational and Network Approaches." In *Proceedings of the 1974 ACM SIGFIDET workshop on Data description, access and control: Data models: Data-structure-set versus relational,* 11–41. Ann Arbor, Michigan: ACM, 1974.

Comer, Douglas. "The Ubiquitous B-Tree." *ACM Computing Surveys* 11, no. 2 (1979): 121–137.

Computer History Museum. Oral history of C. J. Date. Interviewed by Thomas Haigh. Computer History Museum Lot Number X4090.2007, http://www.computerhistory.org/collections/catalog/102658166 (2007).

Computer History Museum. RDBMS Workshop: IBM. Grad, Burton, moderator. Computer History Museum, Lot number X4069.2007, http://www.computerhistory.org/collections/catalog/102702563 (2007).

Cortada, James. *Before the Computer: IBM, NCR, Burroughs, and Remington Rand and the Industry They Created, 1865-1956*. Princeton University Press, 1993.

Cortada, James. *The Digital Hand: How Computers Changed the Work of American Manufacturing, Transportation, and Retail Industries*. Oxford University Press, 2003.

Cortada, James. *The Digital Hand, Volume 2: How Computers Changed the Work of American Financial, Telecommunications, Media, and Entertainment Industries*. Oxford University Press, 2005.

Cortada, James. *The Digital Hand, Vol 3: How Computers Changed the Work of American Public Sector Industries*. Oxford University Press, 2007.

Darwen, Hugh. "The Relational Model: Beginning of an Era." *IEEE Annals of the History of Computing*34, no. 4 (2012), 30-37.

Date, C.J. and Codd, E.F. "The Relational and Network Approaches: Comparison of the Application Programming Interfaces," in *Proceedings of the 1974 ACM SIGFIDET workshop on Data description, access and control: Data models: Data-structure-set versus relational*, 83–113. Ann Arbor, Michigan: ACM, 1974.

Davies, Charles T. "Recovery Semantics for a DB/DC System." In *Proceedings of the ACM Annual Conference*, 136–141. ACM, 1973.

DeCandia, Giuseppe, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. "Dynamo: amazon's highly available key-value store." *SIGOPS Operating Systems Review* 41, no. 6 (October 2007): 205–220.

Desrosières, Alain. *The Politics of Large Numbers: A History of Statistical Reasoning*. Harvard University Press, 1998.

Eisenstein, Elizabeth L. *The Printing Revolution in Early Modern Europe*. 2nd ed. Cambridge University Press, 2005.

Ensmenger, Nathan. The Computer Boys Take Over: Computers, Programmers, and the Politics of Technical Expertise. MIT Press, 2012.

Gray, Jim. "The Transaction Concept: Virtues and Limitations (Invited Paper)." In *VLDB '81: Proceedings of the seventh international conference on Very Large Data Bases*, vol. 7, 144–154. IEEE Computer Society, 1981.

Gray, Jim, and Andreas Reuter. *Transaction Processing: Concepts and Techniques*. Morgan Kaufmann, 1992.

Haerder, Theo, and Andreas Reuter. "Principles of Transaction-Oriented Database Recovery." *ACM Computing Surveys* 15 (1983): 287–317.

Haigh, Thomas. "The Chromium-Plated Tabulator: Institutionalizing an Electronic Revolution, 1954–1958." *IEEE Annals of the History of Computing* (2001) 75–104.

Haigh, Thomas. "Inventing Information Systems: The Systems Men and the Computer, 1950-1968," *Business History Review* 75 (2001) 15–61.

Haigh, Thomas. "A Veritable Bucket of Facts: Origins of the Data Base Management System." *SIGMOD Record*, 35 2006, Nr. 2, 33–49

Haigh, Thomas. "How the Data Got its Base: Information Storage Software in the 1950s and 1960s." *IEEE Annals in the History of Computing* (2009): 7–25.

Hall, Richard H. "The Concept of Bureaucracy: An Empirical Assessment." *American Journal of Sociology* 69, no. 1 (July 1963): 32–40.

Harvey, David. *The Condition of Postmodernity*. Wiley-Blackwell, 1991.

Heide, Lars. *Punched-Card Systems and the Early Information Explosion, 1880-1945* (Johns Hopkins University Press, 2009), 250-251.

Jardine, Donald A., ed. *Proceedings of the SHARE Working Conference on Data Base Management Systems*, Montreal, Canada, July 23-27, 1973 (North-Holland, 1974)

Knuth, Donald. *The Art of Computer Programming,* Vol 3: Sorting and Searching. Addison-Wesley, 1973.

Krippner, Greta R. *Capitalizing on Crisis: The Political Origins of the Rise of Finance*. Harvard University Press, 2012.

MacKenzie, Donald. "Opening the black boxes of global finance." *Review of International Political Economy* (October 2005): 555–576.

Mahoney, Michael S. "The history of computing(s)." *Interdisciplinary Science Reviews* 3, no. 2 (2005): 119–135.

Marx, Karl. *Capital*, vol. 1. Harmondsworth, 1990 [1867].

McGee, W.M. "Generalization: Key to Successful Electronic Data Processing." *Journal of the ACM* 6, no. 1 (January 1959): 1–23.

Mitcham, Carl. *Thinking About Technology: The Path Between Engineering and Philosophy* (University of Chicago Press: 1994)

Nijssen, G. M. “Data Structuring in the DDL and Relational Data Model.” In Klimbie, J.W. and K.L. Hoffman, eds. *Data Base Management: Proceedings of the IFIP Working Conference Data Base Management* (American Elsevier, 1974): 363–384.

Nolan, R. L. “Information Technology Management Since 1960.” In *A Nation Transformed by Information: How Information Has Shaped the United States from Colonial Times to the Present,* edited by A. Chandler and J. Cortada. Oxford University Press, 2000.

Arthur L. Norberg, “High-Technology Calculation in the Early 20th Century: Punched Card Machinery in Business and Government”, *Technology and Culture* 4 (1990): 753–779.

Peirce, Charles S. *Collected Papers of Charles Sanders Peirce*, vol. 2. Belknap Press, 1932.

Perrow, Charles. *Complex Organizations: A Critical Essay*, 3rd ed. McGraw-Hill, 1986.

Pugh, Emerson W. and Lyle R. Johnson and John H. Palmer: *IBM’s 360 and Early 370 Systems*. The MIT Press, 1986

Sassen, Saskia. *A Sociology of Globalization*. W.W. Norton, 2007.

Stinchcombe, Arthur. “Bureaucratic and Craft Administration of Production: A Comparative Study.” *Administrative Science Quarterly* 4, no. 2 (1959): 168–187.

Strauch, Christof. *NoSQL Databases*, 2011. www.christof-strauch.de/nosqldbs.pdf.

Max Weber, *Economy and Society* (University of California, 1968 [1922]),

Williamson, Oliver E. “The Economics of Organization: The Transaction Cost Approach.” *American Journal of Sociology* 81, no. 3 (November 1981): 548–577.

Wooton, Charles W., and Carel M. Wolk. “The Evolution and Acceptance of the Loose-Leaf Accounting System, 1885-1935.” *Technology and Culture* 41, no. 1 (January 2000): 80–98.

Yates, JoAnne: *Control through Communication: The Rise of System in American Management*. Johns Hopkins University Press, 1989

Yates, JoAnne: *Structuring the Information Age: Life Insurance and Technology in the Twentieth Century*. Johns Hopkins University Press, 2008

Tags:Issue Three