

**How Data Got its Base:
Generalized Information Storage Software in the 1950s and 60s**

Thomas Haigh [@computer.org](mailto:tom@computer.org)

[.tomandmaria.com/tom](http://tomandmaria.com/tom)

University of Wisconsin--Milwaukee

**DRAFT FOR REVIEW –
DO NOT QUOTE, CITE OR REDISTRIBUTE**

ABSTRACT: This paper explores the prehistory and origins of Data Base Management System (DBMS) technology. Beginning with the context of administrative data processing in the 1950s it examines the creation of early report generation and file maintenance programs and uses archival material to document their standardization and packaging with the 9PAC and SURGE efforts of the SHARE user group. Shifting to the challenges posed to data processing by the adoption of disks and other random access storage devices in the 1960s it places the development of early DBMS systems such as IMS and IDS in the technological and managerial context of the era. Finally it looks at the work of CODASYL's Data Base Task Group in the late-1960s and early-1970s in applying the "data base" concept to these technologies and defining the capabilities of the DBMS in a surprisingly modern manner as a system providing multiple data interfaces for programmers and ad hoc queries, supporting online and batch operation, giving different applications different views of the same data, and separating data definition and manipulation languages.

INTRODUCTION

The Data Base Management System (DBMS) is the foundation of almost every modern business information system. Virtually every administrative process in business, science or government relies on a data base. The rise of the Internet has only accelerated this trend – today a flurry of database transactions powers each content update of a major website, literature search, or internet shopping trip. Yet very little research addresses the history of this vital technology, or that of the ideas behind it. While some attention has been paid to the DBMS as an important product class for the early software industry, professional historians have given little attention to its technological evolution, influence on data processing practice, or intellectual history.¹ When short technical histories feature in texts and essays by data base specialists these are schematic and generally begin with the publication of a series of CODASYL reports from 1969 onward.² In this article I adopt a different approach, in which

¹ The development of the market for mainframe DBMS systems is explored in M. Campbell-Kelly, From Airline Reservations to Sonic the Hedgehog: A History of the Software Industry. Cambridge, MA: MIT Press, 2003, 145-149&184-191. A short history, focusing on the role of public funding in the emergence of the relational model, is found in National Research Council, Funding A Revolution: Government Support for Computing Research. Washington, DC: National Academy Press, 1999, ch. 6.

² CODASYL was the data processing standards organization best known for its development of COBOL. While many data base textbooks include a few pages on the development of data base theory along with their introductory definitions – for example R. Elmasri and S. B. Navathe, Fundamentals of Database Management Systems. New York: Benjamin/Cummings, 1989 does this well – this can mean

CODASYL's specifications are not the starting point but the culmination of fifteen years of practice and experience by administrative computing specialists. By framing work during the 1950s on report generators and generalized file maintenance systems such as SURGE and 9PAC in the context of the needs and priorities of data processing users I expose the submerged foundations of the DBMS in these earlier, user-driven, projects. Likewise by interpreting work of the 1960s, particularly Charles Bachman's efforts on IDS and other projects, in the context of efforts to build so-called total management information systems I connect the creation of early DBMS systems with the broader managerial and organizational context that drove these efforts. Today CODASYL's data base work is remembered only for its propagation of the network data model. But my final discussion of the work of CODASYL's Data Base Task Group and Systems Committee highlights their role in articulating and disseminating the DBMS concept itself and a surprisingly modern and ambitious description of its architecture and capabilities.

Since the 1950s the storage, retrieval and updating of large volumes of stored data has been a key requirement for most computer applications. This is reflected in the adoption of the term "data processing" as the standard description for administrative computing work from the mid-1950s into the 1980s. Thus no technology has been more vital to the success of the computer as an administrative tool than the development of software to hide the complexities of data manipulation from application programmers and end users. While adoption of DBMS technology posed its own challenges, something I discuss in the companion article written with Tim Bergin, it ultimately cut the cost of application development and made the maintenance and adaptation of administrative systems much easier. During the 1970s the shift of mainstream data processing applications to random access storage and online operation introduced additional complexity which would have overwhelmed most development teams without the adoption of DBMSs and other new kinds of system software such as online transaction processing packages.

Because the DBMS exists behind the scenes, invisible to most people, a few definitions are in order. A modern data base management system is a very complex piece of system software. A single DBMS can manage multiple data bases, each one usually consisting of many different data tables. The DBMS includes mechanisms for application programs to store, retrieve and modify this data and also allows people to query it interactively to answer specific questions. Specialists, known as Data Base Administrators (DBAs) control the operation of the DBMS and are responsible for the creation of new data bases and the definition of the table structures used to store data. One of the most important features of the DBMS is its ability to shield the people and programs using the data from the details of its physical storage. The DBMS polices access to the stored data, giving access only to tables and records for which a given user has been authorized. Because all access to stored data is mediated through the

little when stripped of its historical context. The closest thing to a detailed history is a quarter-century old technical primer J. P. Fry and E. H. Sibley, "Evolution of Data-Base Management Systems" ACM Computing Surveys, vol. 8, no. 1, March 1976, pp. 7-42, pages 19-29. On the technical side, detailed comparisons of early systems are given in C. J. Byrnes and D. B. Steig, "File Management Systems: A Current Summary" Datamation, vol. 15, no. 11, November 1969, pp. 138-142; CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems. New York: [n.p., Distributed by the Association for Computing Machinery], 1971; L. Welke, "A Review of File Management Systems" Datamation, vol. 18, no. 10, October 1972, pp. 52-54; D. B. Steig, "File Management Systems Revisited" Datamation, vol. 18, no. 10, October 1972, pp. 48-51.

DBMS, a data base can be restructured or moved to a different computer without disrupting the programs written to use it.

THE NEED FOR FILE MANAGEMENT IN EARLY DATA PROCESSING

The DBMS evolved from generalized file maintenance and report generation created within the unglamorous world of corporate data processing to simplify the creation of programs for routine administration. File maintenance systems were intended to reduce the cost of producing routine administrative programs, and to make the finished programs easier to change and maintain. Report generation systems made it easier to produce printed reports based on particular criteria.

Computer use in the early 1950s was dominated by scientific computation and modeling applications. But by the end of that decade the balance had shifted decisively toward business administration applications.³ Most computer installations were organized around running a handful of automated clerical tasks, most commonly payroll and accounting, over and over again. During the 1950s and 1960s, the vast majority of corporate data processing jobs were either run through the computer one at a time manually or queued using a simple operating system.

Programs and the input data they were processing were loaded together into the computer. Input data was entered onto punched cards by specialist key-punch operators. On smaller, cheaper computers such as the IBM 650, the workhorse machine of the late 1950s, these cards would remain the main storage medium. On larger computers, such as the IBM 700 series machines the data punched onto the cards was usually transferred to tape for processing and storage.

None of this was, to borrow a phrase from the era, rocket science. Processing a weekly payroll run seems like it should be easier than modeling a missile flight path. But the first generation of American data processing installations spent much more and took far longer than expected to get their electronic marvels doing useful work. General Electric's famous 1954 use of a Univac computer to automate payroll processing set the pattern for other firms.⁴ Data processing managers were shocked by the complexity of programming work and the rigid requirements computer technology imposed on areas such as data entry and the handling of special cases. Data was stored on tape as a sequence of codes, and efficient processing was possible only when the tape was read from start to end with a minimum of rewinding or searching. Early computers had small internal memories, which limited the complexity of each application program and the amount of data it could process. Like punched card machines before them, early computers generally worked on one record at a time. Memory limitations, coupled with the inflexible, serial nature of tape storage, meant that a single major job such as payroll might require dozens of programs to be run one after another, each reading and writing information from several tapes. Each would read intermediate results written during earlier runs.

³ Administrative computing during this era is discussed in T. Haigh, "The Chromium-Plated Tabulator: Institutionalizing an Electronic Revolution, 1954-1958" IEEE Annals of the History of Computing, vol. 23, no. 4, October-December 2001, pp. 75-104.

⁴ R. F. Osborn, "GE and UNIVAC: Harnessing the High-Speed Computer" Harvard Business Review, vol. 32, no. 4, July-August 1954, pp. 99-107.

programmers spent much of their time crafting routines to read records from tapes and print lines on paper, dealing each time with the many errors, synchronization problems, tape jams and so on that could frustrate their task.

File management systems evolved from the reuse of subroutines written to handle input and output tasks within application programs. Early application programs were written in assembly language and included all the instructions necessary to specify the minute details of reading and writing information to and from tape, card, or drum.⁶ The pioneers of the late 1950s and early 1960s developed many new techniques and approaches as they struggled to contain programming and operations costs while maximizing flexibility.

Programming groups soon hit on the idea of producing a single set of well written and reusable subroutines to handle these chores. Standard code was modified slightly to fit the particular situation and then inserted into each application program. Technological change also played a part in pushing data processing teams toward the use of standardized produced routines for input and output. As computer manufacturers began to build more powerful capabilities into their data processing hardware, including buffers and auxiliary processing units to smooth the flow of data, the programming work required to read and write records on tape with maximum efficiency became more complex. For example, IBM's tape hardware for its model 705 large administrative computer (announced 1954) provided some assistance in boosting the productivity of data processing operations. At full speed its tape drives could read 15,000 characters a second either forward or backward, from a reel holding five million characters. A special marker character would cause the tape drive to stop reading when it reached the end of a file. Another special mark signaled the end of each record, simplifying the task of reading one record at a time. A single read instruction would transfer five characters at a time for improved efficiency. Tape units functioned somewhat independently of the main processor, under the control of separate tape control unit. Read and write operations were automatically verified using a parity bit. Specified areas of main memory were used as buffers, so that the computer could work on other tasks while transfers completed. Drives could even read and write simultaneously to the same tape, for efficient file updates. The tape drive also recognized a special instruction to rewind by one unit record, making it easier to repeat a read operation that provided garbled data.⁷ But the application programmer remained responsible for writing code to carry out operations such as parsing a record into fields, searching for a record with a particular key value, or inserting a record into an existing file.⁸

file" of personnel cards with the attendance information punched onto a weekly punched card by an IBM time clock). The concepts of file, record, and field were transferred directly to tape storage – though the records were now laid out sequentially along the strip of magnetic tape. Additional codes were introduced to mark the beginning and end of files and provide checks against corrupted data.

⁶ D. D. McCracken, H. Weiss and T.-h. Lee, Programming Business Computers, New York: John Wiley and Sons, 1959, 178-204.

⁷ International Business Machines Corporation, IBM Electronic Data-Processing Machines Type 705 Preliminary Manual of Operations (22-6627-4) Bitsavers, 1957 [cited 2008 January 18]; at http://www.bitsavers.org/pdf/ibm/705/22-6627-4_705_Oper_Jun57.pdf.

⁸ The 727 magnetic tape unit is discussed in *Ibid.* [cited]. Other specialized control units hooked up to the same tape drives could transfer data between tapes and cards or print without the involvement of the main computer.

Placing metadata into file headers and using standard routines to manipulate records and fields made it easier to reconfigure the details of file formats or hardware configuration without needing to make significant changes to application code. Early application programs were closely tied to particular hardware configurations— even changing the tape drive used for temporary storage required considerable programming work, while adapting a program to make efficient use of more memory or additional tape drives sometimes involved a fundamental rewrite. The problem was compounded as companies attempted to reap the benefits of automation by using the output of one major application as the input to another, for example by linking their production scheduling system to their inventory control system, their accounts receivable system and their billing system. Because one major application might contain dozens of small programs, each reading and writing certain files, it could take Herculean efforts on the part of the programming staff to do something as simple as adding an extra digit to the employee number. Adopting generalized file manipulation code and creating standardized descriptions of the record format used in each file began to make it easier for programmers to modify record formats without completely rewriting programs. (Because COBOL included special support for defining file formats its general adoption from the mid-1960s onward also simplified the programmer labor in these areas).

Another problem was the difficulty in extracting information from the computer – while daily, weekly or monthly runs of different parts of a payroll system might each produce voluminous printed reports, the only way to obtain a special report was to write another program. If a manager needed to tabulate data in a different way, or to include only a subset of the original records in the calculations, he could either wait for a programmer to become available or wade through the printout tallying records manually. By the late 1950s, the more innovative data processing teams had begun to address this through the creation of "report generation" programs, into which programmers fed a descriptions of the output desired and of the organization of the data inside the relevant “master files” and were rewarded with the desired reports.

The work on generalized reporting systems carried out in 1957 by General Electric’s team at the Hanford Nuclear Reservation on its IBM 702 (IBM’s first large computer designed primarily for administrative use) was particularly important in the establishment of these techniques. Generalized file sort routines had already been in use for some time, but the idea was to produce a suite of three new generalized programs: the report generator, an improved sort routine, and a file maintenance program able to make changes to file formats as well as file data. All three worked with a consistent “source file dictionary” describing record formats, which was included as a header at the start of each data file. The report generator program used this data dictionary and an input file holding details of the desired report format (which could include calculated fields such as totals and subtotals) to produce a customized program which, when compiled and executed, would yield the desired output. An updated version of the GE report generator, produced the next year, added integral sort capabilities and expanded support for

complex calculations.⁹ Because it was optimized for memory efficiency the system could produce up to twenty reports simultaneously on a single run through the source data tapes.¹⁰

The objective was to make data stored on tape easily accessible for reporting purposes, just as it was in traditional punched card systems. The urgent need for programs of this kind reflected the complexity of the computer as a general purpose machine. Things which were easy to accomplish on the specialized hardware of punched card machines required elaborate programming to accomplish with a computer. Punched card machines, in contrast, were specialized for specific tasks such as sorting or tabulating and were configured by setting switches or inserting wires into a patch board. Project participant Russell McGee remembers that the Hanford managers responsible for business programming and computer operations “all had extensive punched card backgrounds, which had a profound influence on our work.” The ultimate goal was “a single powerful computer on which all data for the organization... was stored and available for reporting and processing.”¹¹ His colleague Fred Gruenberger noted at the time that punched card experience also shaped customer expectations. Managers who requested changes to reports were “vitriolic” when told that minor report changes required the data processing group to “program it at the cost of \$1,500” when they had been “well trained by us in data processing over the years to the idea that all you have to do to change your report is move five wires on the 405.”¹² Veterans of this effort, including McGee and Gruenberger, went on to prominent positions with other organizations.¹³ The Hanford system received significant publicity, and as discussed later was a direct influence on later work carried out by the SHARE user group.

STANDARDIZED FILE MANAGEMENT SYSTEMS: 9PAC, SURGE, AND RPG

The General Electric effort was not the only one of its kind, and participants quickly realized that companies were wasting effort by producing their own systems. The October 1957 meeting of the IBM SHARE user group included panel organized by Fred Gruenberger devoted to this topic. He noted that development of these systems had so far been a “100 percent non-cooperative programming venture which curiously enough took place at about four independent installations at almost the same time within a period of six months.”¹⁴ He hoped to change this by encouraging collaboration.

⁹ For contemporary documentation of the system see W. C. McGee, "Generalization: Key To Successful Electronic Data Processing" Journal of the Association for Computing Machinery, vol. 6, no. 1, January 1959, pp. 1-23; R. C. McGee and H. Tellier, "A Re-Evaluation of Generalization" Datamation, vol. 6, no. 4, July-August 1960, pp. 25-29. However, the most detailed description is given in R. C. McGee, My Adventures with Dwarfs: A Personal History in Mainframe Computers: Charles Babbage Institute, 2004, 52-59.

¹⁰ SHARE Inc, "SHARE Reference Manual for the IBM 704" 1958, SHARE, Inc. Records (CBI 21), Charles Babbage Institute, University of Minnesota, Minneapolis, 65.

¹¹ McGee, My Adventures with Dwarfs.

¹² SHARE Inc, "SHARE Reference Manual for the IBM 704", 49-50.

¹³ Russell (“Russ”) McGee is not to be confused with Bill McGee, who headed the scientific computing group at Hanford during the same period.

¹⁴ SHARE Inc., "Verbatim Transcript of the 9th Meeting of SHARE, October 3, 1957, Morning Session" 1957, SHARE, Inc. Records (CBI 21), Charles Babbage Institute, University of Minnesota, Minneapolis, 49.

During its first few years in the computer business IBM left systems programming work to its customers. The model 702, the firm's first large administrative computer, was accompanied by nothing more than listings for a symbolic assembler and a sample sort program. The code for both was presented in its programming manual, "for study and actual application if desired." In 1956 there were still only nine IBM programmers working on programming issues for its successor, the 705.¹⁵

IBM did make some effort to cater for the needs of its data processing customers. Among IBM's first supported programs for the 705 were generalized routines to process an input file into a desired output order and to merge records read from two or more input tapes into a single output file. With its conventional punched card machines these operations were easily accomplished by physically shuffling the cards into new sequences. Cutting the magnetic tape into thousands of little pieces and sticking it back together was hardly a viable option, but as tape capacity dwarfed the size of core memory it was not practical to read the entire file into memory. Before a file could be used for a particular task, such as generating required output or being used to update a set of master records, its contents had to be sorted into the appropriate order. IBM's "705 Generalized Sorting Program Sort 51" was issued in 1955. Sorting with this program was an involved process, in which the original file was split into two files. These were repeatedly copied from two input tapes to two output tapes. Each sorting run brought the records a step closer to being ordered, and when just two ordered sequences remained the two working files were consolidated into a single output file. In the worst case a single input tape holding fifty thousand records of one hundred characters each might require seventeen runs through the machine, occupying it for around an hour.¹⁶ (Sort programs sound mundane, but remained an important class of utility software into the 1970s when some of the most important independent software products were generalized sort routines).

But during the mid-1950s IBM's main strategy for providing its customers with ready-made programs was to help them to help themselves. SHARE was intended for users of IBM's large scientific computers, and a corresponding group GUIDE was formed for users of its most expensive administrative machines. It supported the work of the SHARE user group established for users of large

¹⁵ The contributions of this group included a version of the powerful Autocoder assembler. C. J. Bashe, L. R. Johnson, J. H. Palmer and E. W. Pugh, IBM's Early Computers. Cambridge, MA: MIT Press, 1986, 345-347, 355-356.

¹⁶ International Business Machines Corporation, IBM 705 Generalized Sorting Program Sort 51 Bitsavers, 1956 [cited December 10 2008]; at www.bitsavers.org/pdf/ibm/705/32-6831_705_Generalized_Sorting_Pgm_1956.pdf. The logic behind a total of seventeen runs is fifteen sorting runs of half full tapes plus final and initial runs of a full tape. The program was designed to store a maximum of four records in memory at once, which explains both its dismal performance and the relatively large maximum record size of up to 2494 characters in size. A more complex program presented with a smaller maximum record size or running on computers with a larger core memory would have been able to sort short record sequences within core memory, reducing the number of sorting passes required through the tape. Sorting methods of this era are explored in E. H. Friend, "Sorting on Electronic Computer Systems" Journal of the Association for Computing Machinery, vol. 3, no. 3, July 1956, pp. 134-168. Calculating the performance of this routine helps us to appreciate the joy with which improved sorting algorithms, such as C.A.R. Hoare's Quicksort, were greeted by programmers in the first decades of electronic data processing. C. A. R. Hoare, "Quicksort" Computer Journal, vol. 5, no. 1 1962, pp. 10-15.

IBM scientific computers and the GUIDE group for users of its large administrative computers.¹⁷ These groups create program libraries full of routines contributed by user companies, and organized project committees to design and create ambitious new programs flexible enough to meet the needs of many companies. One of the main concerns of GUIDE was the establishment of standard methods of structuring, labeling and accessing data files. Standards were more effective when paired with “libraries” of subroutines or blocks of prefabricated code. A programmer could specify the appropriate parameters and trigger the “macro” feature of the assembler to insert the actual code required.

IBM’s main initial contribution to these efforts to handle the logistics of cataloging and distributing the software libraries, though by the late 1950s it was increasingly involved in contributing programmer effort to implement the packages specified by SHARE. SHARE’s biggest and best known early project was the design of a full suite of system software for the newly announced IBM 709. This project, known as SOS for SHARE 709 System, included an assembler, monitor, job control system, and range of other components. One of the projects was a standard “input-output system” designed to provide basic subroutines for input and output operations on cards or tape.¹⁸ In 1957 designers were already distinguishing between logical and physical aspects of data storage in tape files.¹⁹

During this era SHARE’s existing users were dealing with IBM’s 704, a machine optimized for scientific calculation which was rarely applied to administrative tasks. Its successor, the 709, was promoted as being equally adept at administrative work and technical computing. Companies interesting in using the 709 for administrative work founded the new SHARE Data Processing Committee, which met for the first time on October 2, 1957.²⁰ During the first half of the inaugural Data Processing Committee meeting members spent a lot of time trying to standardize terms such as “card record” and “tape file.”²¹ In the second half, under the chairmanship of Charles W. Bachman of Dow Chemical, they

¹⁷ SHARE is discussed, with particular reference to SOS, in A. Akera, "Voluntarism and the Fruits of Collaboration" Technology and Culture, vol. 42, no. 4, October 2001, pp. 710-736.

¹⁸ Progress on implementation of the I/O package is discussed in J. King, "Progress Report on 709 Input-Output, SSD 017" August 19 1957, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC. SOS eventually included an macro-based “Input System” known as INTRAN and an “Output System” known as OUTRAN as well as support for buffering and transmission of input and output in its memory resident monitor programs. International Business Machines Corporation, SOS Reference Manual (incl Distributions 1 to 5) Bitsavers, 1961 [cited 2008 January 18]; at http://www.bitsavers.org/pdf/ibm/share/SOS_Reference_Manual_Jun61.pdf.

¹⁹ C. Mock, "The MockDonald Multiphase System for the 709, SSD 19" 9 September 1957, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

²⁰ C. W. Bachman, "Report of the SHARE Data Processing Committee, October 2, SSD 020" 1957, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

²¹ This yielded the DP glossary in C. W. Bachman, "SHARE Data Processing Committee: Selected Glossary, SSD 022" November 18 1957, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC. This glossary included the concept of a key as a unique identifier used to locate a record.

“extensively discussed” the report generation system produced by General Electric’s Hanford team, and resolved to use this as a prototype for an “input generator” able to parse input data into a specified file format. They established a subcommittee to work on standardized report and input generators suitable for use with the programming facilities of SOS.²²

This eventually gave rise to two important projects. SURGE, under the leadership of Fletcher Jones (then of North American Aviation, soon to cofound Computer Sciences Corporation), was intended to be a short-term project to build something called the “704 Data Processing Sort Routine” to provide data processing capabilities on the aging IBM 704 computers which still the mainstays of most SHARE installations.²³ The new name reflected an increase in its scope to Sort, Update, Report Generate, Etcetera. While a 704 version did appear, SURGE appears to have evolved into an ambitious data processing compiler project for the IBM 709 and 7090.²⁴ An initial 709/7090 version of SURGE was shipped in 1960 with an improved version planned for 1962. SURGE was intended to be accessible to non-specialist programmers, and used a rigid “check-off” format for coding rather than allowing English-like statements along the model of COBOL.²⁵

The initially more ambitious 9PAC, for 709 PACkage, was intended from the beginning to run on the forthcoming IBM 709. The effort was led by Russell McGee. According to McGee, his data processing group at G.E. Hanford was then planning to shift its work from an IBM 702 to the newer and more powerful IBM 709. As the idea of a range of compatible computers was not yet invented, this shift required reprogramming every application. “Only the generalized routines made recommendation of the

²² The core programming system in SOS was called SCAT, for SHARE Compiler-Assembler-Translator. Data Processing committee members monitored its progress closely, providing feedback on the suitability of its input-output capabilities for their purposes. C. W. Bachman, "SHARE Data Processing Committee Meeting, November 21-22, 1957, Midland, Michigan, SSD 023" 1957, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

²³ Data processing committee member William Orchard-Hays, better known for his work on mathematical programming, wrote in 1959 that the “704 Data Processing Sort Routine... is now known as SURGE.” W. Orchard-Hays, "Letter to Bill Dobrusky, March 16, SSD 049" 1959, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC. SURGE has rather a low historical profile, and citations in surveys are generally to an undated document “SURGE: A Data Processing Compiler for the IBM 704” issued by North American Aviation. This may be the same document archived as R. Paul and W. Davenport, "Untitled SURGE Manual for IBM 704 (Title page missing?)" Circa 1959, SHARE, Inc. Records (CBI 21), Charles Babbage Institute, University of Minnesota, Minneapolis.

²⁴ Early objectives of and specification for the 709/7090 SURGE project are discussed in B. Went, "Minutes of 709-7090 SURGE Meeting, September 14-16, 1959 - Columbus Ohio, SSD 59" 1959, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC and E. Austin, "Minutes of 709-90 SURGE Subcommittee Meeting, December 7-9" 1959, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

²⁵ L. F. Longo, "SURGE: A Recoding of the COBOL Merchandise Control Agreement" Communications of the ACM, vol. 5, no. 2, February 1962, pp. 98-100.

709 feasible. The plan was to re-implement the generalized routines for the 709 and the same source files and packets could be used on the new machine as on the old."²⁶ But reworking the generalized routines for the 709, with its more powerful and complex tape control system, was itself a major undertaking. SHARE Data Processing committee members had learned about the Hanford effort in October, 1957 when a report entitled "Generalization: Key to Successful Electronic Data Processing" was distributed. At the next meeting, in November, McGee appeared to give an update on the project's progress and report that an improved version was now operational and that a rewrite for the 709 was planned.²⁷ This reimplementation effort soon became a SHARE project, a transition McGee recalls took place at the tenth SHARE meeting in Washington DC the next February.²⁸

The SHARE development process had a great deal in common with today's open source software projects. A small volunteer committee handled the main development and specification tasks, while a series of SHARE Secretarial Distributions sent to all IBM 709 sites kept the broader user community informed. These distributions include much discussion of the project, including a series of drafts of documentation, requests for information, comments and suggestions. Starting a project was easy. As McGee recalls "creation and dissolution of these committees and subcommittees were accomplished by simple word-of-mouth agreements between the chairpersons and the SHARE officers."²⁹ Formalities could lag practice. SHARE documents show that the 9PAC Subcommittee of the SHARE Data Processing Committee was formally chartered, with McGee as its chair, only after a meeting of interested parties in Los Angeles in May 1959 when the project was already well underway.³⁰ The hard part was persuading SHARE member companies to devote programmer time to the effort, which McGee accomplished through "frequent trips to various companies to tell my story to the brass." He succeeded in persuading Union Carbide to provide office space to coordinate the Report Generator project in its Long Island City facility.

According to a progress report dated November 1958 the report generator was partially operational and "While the SHARE committee has been forging ahead with completion of the report generator, the generalized file maintenance program, which was undertaken as a SHARE project by General Electric, Hanford, has been making progress. At the present time, there are nine people assigned to the project. We are estimating that all coding of file maintenance will be completed in December [1958] and that the

²⁶ McGee, My Adventures with Dwarfs, 59.

²⁷ Bachman, "SHARE Data Processing Committee Meeting, November 21-22, 1957, Midland, Michigan, SSD 023".

²⁸ McGee, My Adventures with Dwarfs. SHARE minutes report that McGee made another progress report, and also note his chairmanship of a newly formed "709 Report and File Maintenance Generator Subcommittee." C. W. Bachman, "Report of SHARE Data Processing Committee" in, SHARE Inc, Ed., Proceedings of the Tenth Meeting of SHARE, Washington DC, February 26-28, 1958, pp. 51-55.

²⁹ McGee, My Adventures with Dwarfs, 60.

³⁰ R. C. McGee, "Letter to Jerry Koory, SSD 054" June 23 1959, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

system will be in operation by the first of February.”³¹ Hanford’s own 709 was not yet ready for use, and a debugging trip to an installation in Los Angeles in January revealed “many errors” which McGee did not expect to be able to correct until the end of that month when they should have to 709 time at Hanford and could reassemble the code.³²

Debugging took a little longer than McGee expected, though less time than we might predict, and 9PAC entered use in May 1959. While IBM had not contributed to its creation, the firm quickly assumed responsibility for maintaining and updating the system. McGee dates this transfer to the summer of 1960.³³ By 1961 an updated version for the IBM 7090 (the transistorized successor of the 709) was “currently being maintained and improved by IBM Applied Programming.”³⁴ 9PAC became an official part of IBM’s sprawling IBSYS operating system suite for the 7090/7094.

Like the earlier Hanford Report Generator, 9PAC took as its input a set of input parameters and file definitions and produced a program which, when run, would do the required job. But there was at least one vital difference between the two systems. Although 9PAC was used exclusively with tape storage, it did permit the creation of hierarchical relationships between records. Child records were simply stored on tape immediate following their parent records. This was a continuation of punched card practice, in which this could be accomplished by sorting appropriately – as long as the same part of the card was always used to code for customer number then it would be easy to use sort and merge operations to create a card file in which each customer record appeared in order of its name and was followed immediately by cards giving information on each order placed by that customer.

The influence of the earlier system is clear. Data definitions for the file were stored in a header. This included field information for records on each level of the hierarchy. 9PAC consisted of two separate but compatible modules: a Report Generator and a Generalized File Maintenance system. The latter incorporated capabilities for calculated updates and modifications to the format of existing files and was implemented by the GE Hanford team.³⁵ The file format was surprisingly complex, allowing storage of

³¹ R. C. McGee, "Progress Report on Generalize Routines, SSD 040" November 7 1958, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

³² R. C. McGee, "Letter to Charles E Thoma, January 21, SSD 046" 1959, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

³³ The date is from McGee, My Adventures with Dwarfs, 64.

³⁴ International Business Machines Corporation, IBM 7090 Programming Systems SHARE 7090 9PAC Part 1: Introduction and General Principles (J28-6166) Bitsavers, 1961 [cited 2008 January 18]; at http://www.bitsavers.org/pdf/ibm/7090/J28-6166_9PAC_Part1_1961.pdf.

³⁵ Specifications for the file maintenance system are in R. C. McGee, "Preliminary Manual for the Generalized File Maintenance System, SSD 046" December 23 1958, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC. Their shared file structure is described in R. C. McGee, "The Structure of a Standard File, SSD 045" November 10 1958, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC. A header was used to store file information, according to C. W. Bachman, "Oral History Interview by Thomas Haigh, 25-26 September 2004, Tucson, AZ" 2004, ACM Oral History Interviews

data dictionary information on a separate dictionary tape or in a header affixed to the beginning of each file. It also allowed the storage of access permission information to restrict the reading or modification of the file.³⁶

SHARE members hoped to create still more complex file management systems. 9PAC and 709 SURGE were seen as short term solutions, stop gaps until ambitious goals were fulfilled. SURGE, for example, was originally announced as “an interim 709-7090 commercial data processing system” for use from mid-1960 until “INGEN and/or the CT [IBM’s Commercial Translator or COMTRAN] are available.”

By August 1958 a subcommittee of the SHARE data processing committee had already drawn up plans for a full “data processing package” for the IBM 709. The specification described a multiprogrammed system, in which a permanently resident task scheduler juggled multiple simultaneous file processing applications and provided communication between them. The system was to provide instructions for file input and output, and when given appropriate field descriptions could extract, store, convert or move information from “fields addressed symbolically by the program.”³⁷ Work on this system does not seem to have progressed further.

Another never-implemented proposal produced during 1959-60 described the INGEN system, a successor to 9PAC which would integrate its report, file maintenance and sort program generation capabilities into a single system able to generate programs combining these three operations. This effort was led by Charles Bachman. INGEN stood for Integrated Generator. According to Bachman, its main advance would have been to make it much easier to reorganize record hierarchies. A hierarchical tape system like 9PAC might structure its order file by customer, following each customer record with a series of orders for that customer sorted by date. A user might want to produce a report listing all orders by date, which would require producing a file holding the same data but structured to have date rather than customer as the top level of the hierarchy. INGEN would have performed this kind of restructuring automatically, based on a metadata system which understood that a particular record type might be a master record in one context and a detail record in another.³⁸

As far as I am aware, none of these plans for next generation tape based file management and report generation systems came to fruition. They were stymied by the inherent limitations of serial access storage and the rapid adoption of disk technology during the early 1960s by the kind of high end computer centers that might have contributed to their development or benefitted from their use. But the key requirements identified for these systems were the same needs that drove the development of data base management systems in the years that followed. Chief among these were support for multiple concurrent applications with a single resident data handling system and the flexibility to situate the same set of records in different record hierarchies depending on the job at hand.

collection. ACM Digital Library. McGee’s memoir provides more detail and confirms that Hanford tackled the sort and file maintenance sides of 9PAC. McGee, My Adventures with Dwarfs, 62.

³⁶ McGee, "The Structure of a Standard File, SSD 045".

³⁷ J. T. Horner, "709 Data Processing Package, SSD 035" July 24 1958, SHARE Records, 1955-86, NMAH 567, Archives Center, National Museum of American History, Behring Center, Smithsonian Institution, Washington, DC.

³⁸ C. W. Bachman, "INGEN Proposal" May 1 1959 or 1960, Charles W. Bachman Papers (CBI 125), Charles Babbage Institute, University of Minnesota, Minneapolis. Bachman clarifies INGEN’s face in Bachman, "Oral History Interview by Thomas Haigh".

FILE MANAGEMENT FOR THE MASSES – RPG AND MARK IV

It wasn't just the users of multi-million dollar computers like the IBM 709 that recognized a need for standard file management and report generation systems. IBM addressed the challenge directly when launching its 1401 computer, the successor to the humble 650 and the workhorse of second-generation data processing during the first half of the 1960s. As the first computer marketed as a viable alternative to conventional punched card technology for mainstream punched card installations, the 1401 had to be easy to use for these simple tasks. IBM supplied a new programming system called Report Program Generator (RPG) for this purpose.

RPG took as its input four decks of cards, each using a simple programming notation to describe the data file format, the desired records and fields to be included from that file, desired report format, or calculated fields (such as interest accumulated). It generated an efficient program which, when executed, would produce the specified report. RPG's inputs mimicked the kind of configuration work formerly handled by wiring the control boards of punched card machines, though the calculation facilities were more flexible than those offered with regular punched card equipment. RPG worked with input on punched cards, in tape files, or (by the mid-1960s) on disk. Its simplest version was usable on a computer with only four thousand words of memory. Like 9PAC, RPG could process files in which master records (for example customers) were followed by the relevant detail records (for example orders placed). When translating this input to a report, RPG could insert headings and subtotals as appropriate.³⁹ RPG was an enormous success, and was soon offered for other machines including the larger 7070 computer. Even today, in somewhat improved versions, it is relied on by many thousands of corporate programmers.

File management systems also proved an important niche for the nascent independent software package industry. Mark IV – the most successful product of the early independent software industry – was a file management system descended from report software produced for the Douglas Aircraft Company. The origins of Mark IV (in a series of systems known as GIRLS, Mark I, Mark II, Mark III) have been discussed in published work by its creator John A. Postley and so need not be recounted at length here.⁴⁰ In this context the most relevant observation is that the successful launch of Mark IV as a standard

³⁹ The discussion of RPG's capabilities is based on International Business Machines Corporation, Report Program Generator: IBM 1401 Card and Tape Systems, J24-0215-2 Bitsavers, 1965 [cited 2008 January 24]; at http://www.bitsavers.org/pdf/ibm/140x/J24-0215-2_cardTapeRPG.pdf and International Business Machines Corporation, Report Program Generator (On Disk) Specifications, IBM 1401, 1440, and 1460, C24-3261-1 Bitsavers, 1964 [cited 2008 January 24]; at http://www.bitsavers.org/pdf/ibm/140x/C24-3261-1_1401_diskRPG.pdf. Because of its apparently mundane nature, RPG has received a lot less historical attention than its usage would justify. Discussion in the secondary literature seems limited to M. Campbell-Kelly and W. Aspray, Computer: A History of the Information Machine. New York, NY: Basic Books, 1996, 133 and Bashe, Johnson, Palmer and Pugh, IBM's Early Computers, 479-480.

⁴⁰ On Mark IV see J. A. Postley and H. Jakobsohn, "The Third Generation Computer Language: Parameters do the Programming Job" in Data Processing, vol. 11, Data Processing Management Association, Ed., Los Angeles, California, 1966, pp. 408-415, R. L. Forman, Fulfilling the Computer's Promise: The History of Informatics, 1962--1982. Woodland-Hills, CA: Informatics General Corporation, 1984 and J. A. Postley, "Mark IV: Evolution of the Software Product, a Memoir" IEEE Annals of the History of Computing, vol. 20, no. 1, January-March 1998, pp. 43-50.

product, which took place in 1968, rested on three years of development worked funded by a small group of initial users and that this, in turn, relied on experience producing similar custom systems under contracts going back to 1960.

RANDOM ACCESS, REAL TIME, AND THE THIRD GENERATION

These file management techniques were very useful with tape storage, but when firms began to start storing their data on disk drives, the extra complexity of programming random access data storage and retrieval made their use almost essential.

Disk storage was called “random access” because information stored on any part of the disk could be rapidly retrieved. This contrasted with the sequential nature of tape storage, where it was often necessary to play through the entire tape to find a specific record. Random access had been around since the early 1950s, in the shape of magnetic drums. As Arthur Norberg has shown, this technology was pioneered by Engineering Research Associates in 1948 under contract to the Navy.⁴¹ IBM’s 650 relied on a magnetic drum for its internal memory, a cost cutting feature. Drums drove early interactive business systems such as the Univac File Computer and American Airlines’ Magnetronic Reservoir, and high speed magnetic drums were a key feature of real time systems like the famous SAGE air defense system. Early magnetic disks were slower than drums, but they were cheaper and stored much more data.

With tape based data processing applications users would still need to keep paper files, or leaf through big piles of routine printout, to get speedy access to a specific record. Disk drives, however, offered “random access” storage, giving almost instant access to any part of a disk. This promised to allow the speedy retrieval of specific data as needed, making it much easier to create special reports or to build on-line business systems. The 305 RAMAC (Random Access Method of ACCounting) introduced by IBM in 1957 was the first computer to use disk storage. RAMAC bundled a simple vacuum tube computer with five million characters of storage (roughly equivalent to three standard floppy disks of the late-1990s) spread over fifty discs spun at 1,200 revolutions per minute. IBM eventually offered the same technology for its mainstream 1401 computer as the 1405 Disc Storage Unit.

The RAMAC itself was a freestanding product, with simple programming capabilities and an old-fashioned control panel that was rewired to configure the machine. It was queried by feeding special trigger cards into it. RAMAC could be integrated into existing punched card operations, and while it was not particularly fast it could process a transaction every one and a half seconds, which was about as quickly as its printer could produce the resulting output. For the first time it was practical to build an administrative application in which a data file was updated continually and could be analyzed as needed. Square D, of Milwaukee, used a prototype model to store stock levels for each item in a 24,000 inventory in its Industrial Controller division. Different decks of cards represented changes in desired stock levels and order quantities, as well as bills of materials to alert the system of new orders received. Another user of RAMAC technology, American Bosch, had switched to “continuous flow” accounting for its production and inventory control, in which transactions were posted as they arrived in the accounting office. Its operators could request special reports at any time by punching special codes onto a “trigger deck” of cards and feeding it into the system.⁴²

⁴¹ A. L. Norberg, Computers and Commerce: A Study of Technology and Management at Eckert-Mauchly Computer Company, Engineering Research Associates, and Remington Rand, 1946-1957 Cambridge, MA: MIT Press, 2005.

⁴² W. L. Jerome and L. Hartford, "RAMAC at Work" Systems and Procedures, vol. 8, no. 4, November 1957, pp. 30-38 and Anonymous, "New Accounting Concept Based on 'Assembly-line' Processing"

RAMAC was real-time (if the term is interpreted broadly), but not on-line. Although it could keep totals continually up to date, it still required all transactions to be punched onto cards. RAMAC's production run, around a thousand machines, was much larger than the number of special-purpose on-line administrative systems produced during the same 1957-1962 period.

The largest and best known of these systems, SABRE, supported reservations for American Airlines. Constructing this system with early-1960s technology was hugely expensive. The first full version relied on two high-end 7090 mainframes, six of IBM's fastest drum units (an expensive niche product then used largely in military systems) and thirteen of its larger but slower 1301 disk units. The control units for operators were still specially built (at a cost of \$19,000 each). In 1963 a typical 7090 configuration was worth about \$2 million, whereas the \$30 million that American paid for SABRE failed to come close to covering IBM's costs.⁴³ Unlike batch operation, where a variety of different jobs were scheduled and run in succession, real-time operation demanded the commitment of an entire computer on a full-time basis. While such systems were not usually available around the clock, the night-shift was occupied with backups, the loading and unloading of data, report generation, compression, upgrades, and testing.⁴⁴

The disk drive was first offered as a standard option for most major computer systems in 1962, though it had been available in a handful of IBM systems a little earlier.⁴⁵ Whereas tape had previously been the only way of magnetically storing large files of information, it was suddenly possible to hold up to one billion characters of data on the disk drives connected to a single large IBM computer.

Disk technology progressed rapidly, and by the mid-1960s disks were standard options on many of the newly announced "third generation systems", along with operating systems, large memories, remote terminals, and other features marketed as the key to on-line application development.⁴⁶ The third generation machines represented a fundamental departure from earlier practice in two key areas: the introduction of the operating system as an integral part of the computer installation, and the addition of

Management and Business Automation, February 1961. Use of the RAMAC is also discussed in R. H. Gregory, "Preparation for Logic -- An Orderly Approach to Automation" in Management Control Systems, D. G. Malcolm and A. J. Rowe, Ed., New York: John Wiley & Sons, Inc., 1960, pp. 169-183 and Anonymous, "Programming An Information Explosion" Business Automation, vol. 14, no. 5, May 1967, pp. 47-50.

⁴³ SABRE is discussed in R. W. Parker, "The SABRE System" Datamation, vol. 11, no. 9, September 1965, pp. 49-52; D. G. Copeland, R. O. Mason and J. L. McKenney, "SABRE: The Development of Information-Based Competence and Execution of Information-Based Competition" IEEE Annals of the History of Computing, vol. 17, no. 3, Fall 1995, pp. 30-57.

⁴⁴ On the programming of real-time systems in this period, and its relationship to hardware features, see W. L. Frank, W.H. Gardener and G. L. Stock, "Programming On-Line Systems. Part Two: A Study of Hardware Features" Datamation, vol. 9, no. 6, June 1963, pp. 28-32.

⁴⁵ E. Webster and N. Statland, "Instant Data Processing" Business Automation, vol. 7, no. 6, June 1962, pp. 34-36, 38; N. Statland and J. R. Hillegass, "Random Access Storage Devices" Datamation, vol. 9, no. 12, December 1963, pp. 34-45; Anonymous, "Disc File Applications: Reports Presented at the Nation's First Disc File Symposium" in, vol., Ed. Eds., ed. Detroit: American Data Processing, Inc., 1964

⁴⁶ E. W. Pugh, L. R. Johnson and J. H. Palmer, IBM's 360 and Early 370 Systems. Cambridge, MA: MIT Press, 1991.

standard hardware for interactive, real-time operation. These were intended to take the achievements that had required so much carefully crafted assembly language and specially built hardware to accomplish for a system like SABRE and make them into a standard capability of the hardware and software installed in a typical large data processing center.⁴⁷

Disk drives typically served as a means of making the computer a more effective tools to tackle traditional administrative and accounting operations. A large disk system promised a single repository into which business data could be placed and from which they could be checked, retrieved and updated by many different application programs. Random access storage was a boon to programmers of many kinds of application. Together with the larger memories of the newer computers it heralded a gradual shift away from the need for dozens of separate “runs” through master files to perform routine updates. But random access storage also demanded a whole new set of programming techniques, analysis methods and conventions.⁴⁸

Random access promised almost instant record retrieval, but although it was easy to order the computer to read a particular part of a disk (such as drive 4, platter 5, side 1, track 3, sector 15), there was no easy way to jump straight to a particular record (e.g. customer account 15274). One could, of course, keep the records sorted in order, but this would require an enormous amount of work rearranging the existing records every time a new one was added. Programmers experimented with a variety of strategies to arrange and index data on random access devices.⁴⁹ No single technique was suitable for all situations, and most of them were very complicated to program.

Another set of problems was caused by having several programs share a single disk, each using different program code to read and write records. Among these problems were the risk that an errant program might scramble an area of the disk holding information belonging to another, the overhead imposed by writing several different versions of the code required to handle complex indexing techniques, and the certainty that at some point the physical layout of the disk storage would be changed (for example to

⁴⁷ Communications generally, and timesharing in particular, were an area of weakness for IBM's original 360 series computers compared with competing models from RCA and General Electric. Fortunately for IBM, although these capabilities were much discussed during the 1960s they were not much user until the 1970s, by which time the deficit had been remedied with the 370 series.

⁴⁸ For example, although a programmer could now retrieve the contents of any of the millions of storage locations on a disk, this was of little use if it was still necessary to comb through every one of those locations to find the record she was after. Some kind of index or table of contents was required – so that the program could first determine where the specific record was stored and then find the data. But nobody had experience producing such a system, and any method adopted would add substantially to the complexity of the program, the storage space needed on the disk and the work required to insert, delete or update particular records. New headaches arose when it was necessary to make a backup copy of the disc system, alter the format in which records were stored, or to change (expand, shrink, or move) portion of the disk system allocated to a particular program. McCracken devoted some consideration to the problems of random access programming in McCracken, Weiss and Lee, Programming Business Computers. For its evolution see R. H. Buegler, "Random Access File System Design" Datamation, vol. 9, no. 12, December 1963, pp. 31-33 and R. G. Canning, "New Views on Mass Storage" EDP Analyzer, vol. 4, no. 2, February 1966.

⁴⁹ A good idea of the techniques available to ordinary programmers faced with early random access storages systems is provided in McCracken, Weiss and Lee, Programming Business Computers, ch. 19.

shift a growing file to its own disk and expand the storage areas for the remaining ones) and all the programs would have to be modified at once.

The complexity of building online applications and working with random access storage held back the spread of these technologies into mainstream use during the 1960s. A survey of managers with responsibility for computer purchasing by the Wall Street Journal in 1968 gave a detailed picture of data processing operations among 634 companies. Despite considerable attention given to timesharing and communications during the period, most computing was still performed in batch mode.⁵⁰ That same year, the long-running survey of computer installations carried out by Business Automation magazine revealed that the new peripheral devices associated with third generation computing were also spreading quite slowly. Disc drives had arrived in just 44 percent of data processing installations. The only advanced storage device to have reached most installations was the tape drive – the punched card processing era was finally winding down, as just 28 percent were still without a single tape drive, while the median installation had four. Punched cards retained their supremacy for data input, however, with 85 percent of the departments still reliant on the simple keypunch as the main source of data. The video display, then emerging as the symbol of cutting-edge computer had reached just 13 percent of installations.⁵¹

IDS AND IMS

Computer vendors aimed to help their customers manage this enormous increase in complexity with a new breed of generalized file management systems built to work with random access discs. By the end of the 1960s, every major computer manufacturer offered at least one piece of advanced file management software with support for disk storage.⁵² They were usually based on the expansion of systems originally produced for use within a single organization. These systems were intended to speed program development, reduce maintenance costs, shield application programs from the consequences of changes in the physical disk layout and make it easier to selectively retrieve records based on their contents.

The most innovative, and influential, of these systems was General Electric's Integrated Data Store (IDS), created by Charles W. Bachman. IDS began life circa 1963, as part of an effort known internally as "Integrated Systems Project II." Its goal was the production of an integrated system for production control, flexible enough to be easily customizable by GE's many departments but powerful enough to give rapid results to queries on production scheduling and inventory levels while automatically placing orders and calculating the optimum order quantities. The resulting system, MIACS (sometimes, but not always, Manufacturing Information And Control System) relied on IDS to handle its data storage and retrieval needs.⁵³ In the early 1960s many companies launched ambitious efforts to produce integrated

⁵⁰ Wall Street Journal, "Management and the Computer: A Wall Street Journal Study of the Management Men Responsible for their Companies' Purchases of Computer Equipments and Services" 1969, Data Processing Management Association Records (CBI 88), Charles Babbage Institute, University of Minnesota, Minneapolis.

⁵¹ Anonymous, "EDP Salary Survey--1969" Business Automation, vol. 16, no. 6, June 1969, pp. 48-59.

⁵² Developments in this area were analyzed in Canning, "New Views on Mass Storage" and R. G. Canning, "Data Management: File Organization" EDP Analyzer, vol. 5, no. 12, December 1967.

⁵³ An early description of IDS in the context of the integrated systems project is given in C. W. Bachman, "GEICS - General Electric Integrated Computer System" n.d., Charles W. Bachman Papers (CBI 125), Charles Babbage Institute, University of Minnesota, Minneapolis. IDS is described in a

systems tying together different business functions.⁵⁴ Thanks to Bachman's ingenuity, the MIACS project was a rare success in this area. These systems were often called MIS, for Management Information Systems. Bachman recalls that top management were told that the project name stood for Management Information And Control System.⁵⁵

Manufacturing involves the assembling of multiple components into larger parts, which themselves usually serve as components in one of more kinds of larger assemblage. The need to solve this "parts explosion" problem made it particularly important for IDS to support the creation of linkages between different kinds of record. While earlier systems had supported the idea of sub records, stored sequentially and hierarchically within master records, IDS was much more flexible. This generalized concept of linkages between record types, known later as the "network data model" was a major influence on early DBMS implementations.⁵⁶

IDS was designed from the beginning for use with disk drives. It took over an entire GE 225 computer, providing basic operating system functions including an early implementation of paged virtual memory to squeeze out maximum performance from computer's 8 thousand word standard memory. The task scheduler for the MIACS application itself relied on IDS to store data. MIACS application programs (written in General Electric's own GECOM language) used simple instructions to navigate through the relationships between records and to STORE, GET, MODIFY or DELETE records one at a time. In the first implementation of IDS, a preprocessor replaced these special instructions with the appropriate strings of assembly instructions. However, efficiency concerns forced a switch to a different approach, where IDS performed this expansion interpretatively, combining the requested operation with metadata about the record type involved. This part of IDS remained resident in memory, waiting to deal with data requests from the application programs.⁵⁷

companion article by Bachman in this issue, and in a number of the manuals and design documents preserved in his archival papers at CBI. Published descriptions include C. W. Bachman, "Integrated Data Store" DPMA Quarterly, vol. 1, January 1965, pp. 10 and CODASYL Systems Committee, Survey of Generalized Data Base Management Systems, May 1969. New York: Association for Computing Machinery, 1969, sect. 5.

⁵⁴ See T. Haigh, "Inventing Information Systems: The Systems Men and the Computer, 1950-1968" Business History Review, vol. 75, no. 1, Spring 2001, pp. 15-61.

⁵⁵ Bachman, "Oral History Interview by Thomas Haigh".

⁵⁶ IDS is often, and with some justification, called the first data base management system. On a technical level, it was years ahead of its time and pioneered many important techniques. Powerful as it was, however, the initial version of IDS lacked some of the features associated with later systems and formalized in the CODASYL definition of a DBMS. Record definitions were punched directly onto cards in a special format rather than being defined and modified via a data definition language. It did not provide an interface for ad-hoc querying, or support for online access, since it was created purely to support the MIACS application. It did not provide different views or subsets of the overall database to different users. Neither did it support multiple databases simultaneously. Just as importantly, nobody at the time called it a data base management system. That concept itself did not exist until around 1968.

⁵⁷ Bachman, "Oral History Interview by Thomas Haigh".

IDS was used at a number of sites internally within General Electric. An optimized version was issued by International General Electric for its 225 system and supplied to customers including Mack Truck and Weyerhaeuser. With the advent of the third generation GE 400 and 600 series computers a new version of IDS was produced, in which application programs were written in the now standard COBOL rather than GE's own language. These versions were produced by the systems programming groups for the new machines. These standard IDS implementations lacked transaction processing capabilities – IDS was loaded and unloaded from memory along with the application programs using it.

A special version of IDS was in use at Weyerhaeuser, in its WEYCOS (Weyerhaeuser Comprehensive Operating System). The first version of WEYCOS handled inventory control and order processing. It worked on a GE 225 in Tacoma, Washington which was hooked up to a Datamet 30 computer. This interfaced with teletype machines across the country, accepting input data and requests for reports directly from sales offices, mills and warehouses. Requests were dumped onto disk and processed by IDS's integral task scheduler (the "problem controller"). Participants recall that this system was fully operational by 1965. A more ambitious system, which Bachman calls WEYCOS 2, was developed for the GE 600 computer from 1966 to 1968. This, he believes, was the first "multiprogramming database system" because it allowed multiple application programs to run simultaneously, each executing its own transactions against a shared database. This meant adding capabilities to lock records and detect deadlock conditions. The system was also intended to support multiple processors with shared memory, though this goal was eventually abandoned.⁵⁸

A few years later, around 1965, the first version of what eventually became IBM's Information Management System (IMS) was produced by IBM in collaboration with North American Rockwell to handle the proliferation of parts involved in the Apollo program.⁵⁹ The original version of this application, known as Generalized Update Access Method, ran on an IBM 7010 computer, and used a specialized hierarchical file management system to store its data on disk. IBM and NAA also developed a system called RATS (Remote Access Terminal System) so that interactive application programs could be accessed via terminals. In 1966 work began on a new version created to run as an application under OS/360 on the new System 360 machines, and it was this version that IBM distributed to other customers from 1968 onward.⁶⁰ Like IDS, IMS was used by application programmers, using packaged procedures to embed data handling capabilities in their code. The OS/360 version allowed one memory resident copy of IMS to simultaneously service the data needs of multiple application tasks.⁶¹

Although it went on to great commercial success in the 1970s, IMS was not the only IBM data management product of the late 1960s. In the mid-1960s members of the SHARE user group devoted far

⁵⁸ *Ibid.*, 82-92 and McGee, My Adventures with Dwarfs, 99-110.

⁵⁹ K. R. Blackman, "IMS Celebrates Thirty Years as an IBM Product" IBM Technical Journal, vol. 37, no. 4 1998, pp. 596-603.

⁶⁰ An excellent contemporary introduction to the first public version of IMS is given in J. W. Adams, "IMS - Information Management System/360" in Proceedings of SHARE 31, Oct. 28-Nov. 1: SHARE, Inc., 1968, vol. 2 231-263.

⁶¹ R. L. Patrick, "Oral History Interview with Thomas Haigh, February 16 2006, Mountain View, CA" 2006, Forthcoming from Computer History Museum, Mountain View, CA, Charles Babbage Institute, Software History Dictionary Project: Information Management System (IMS) Charles Babbage Institute, February 17 2004 [cited April 15 2006]; at <http://www.cbi.umn.edu/shp/entries/ims.html>.

more time in their sessions and workshops to discussing the forthcoming GIS, for Generalized Information System. This was expected to support file manipulation, reporting, full text indexing, interactive querying and batch updates. Its actual capabilities were the subject of much speculation prior to its release.⁶² While GIS was part of IBM's product line in the 1970s its actual use seems to have been as a query language and it did not achieve much success as a freestanding product.

General Electric offered an improved version of IDS to users of its computers, and IBM did the same with IMS. During the 1960s computer vendors "bundled" their software with hardware, using it as a free promotional tool to entice users into buying computers.

SDC AND THE DATA BASE CONCEPT

IDS and IMS both predated the idea of the "data base management system" or DBMS. They did not, however, predate the data base concept itself which originated around 1960 but was initially quite separate from file management technology. My earlier article "A Veritable Bucket of Facts" explored this process, so I shall not dwell on its details here.⁶³ The data base concept originated among the well-funded cold war technologists of the military command and control world. It originated in or before 1960, probably as part of the famous SAGE anti-aircraft command and control network. SAGE was far more complex than any other computer project of the 1950s, and was the first major system to run in "real-time" – responding immediately to requests from its users and to reports from its sensors.⁶⁴ As a result, SAGE had to present an up-to-date and consistent representation of the various bombers, fighters and bases to all its users. The System Development Corporation a RAND Corporation group spun-off to develop the software for SAGE, had adopted the term "data base" to describe the shared collection of data on which all these views were based.

SDC invested heavily in this area and identified "computer-centered data base systems" as a key application of time-shared systems – hosting (in collaboration with military agencies) two symposia on the topic in 1964 and 1965.⁶⁵ These were crucial in spreading the data base concept to high-ranking military officials, business data processing celebrities, and corporate and academic researchers. Reporting on the event in *Datamation*, the leading trade magazine of business computing, Robert V. Head observed that data bases had already unleashed the "biggest single strike" of new jargon "since the great time-sharing goldrush of 1963," leaving potential users "sullen and down-trodden." He concluded

⁶² Many sessions were devoted to GIS at SHARE 27 in 1966 and 28 and 29 in 1967. By 1969 it had at least some actual users – see J. F. Fry, "Recent User Applications with GIS" in Proceedings of SHARE 33, Aug. 18-22 1969, S. Inc, Ed., 1969, pp. 295-6. Its early capabilities are summarized in CODASYL Systems Committee, Survey of Generalized Data Base Management Systems, May 1969, sect. 4.

⁶³ T. Haigh, "'A Veritable Bucket of Facts': Origins of the Data Base Management System" in Proceedings of the Second Conference on the History and Heritage of Scientific Information Systems, M. E. Bowden and B. Rayward, Ed., Medford, NJ: Information Today Press, 2004.

⁶⁴ SAGE is discussed in P. Edwards, The Closed World: Computers and the Politics of Discourse in Cold War America. Cambridge, MA: MIT Press, 1996 and T. P. Hughes, Rescuing Prometheus. New York: Pantheon Books, 1998.

⁶⁵ See Anonymous, "A panel Discussion on Time-Sharing" Datamation, vol. 10, no. 11, November 1964, pp. 38-44 and System Development Corporation, "Preprint for Second Symposium on Computer-Centered Data Base Systems, Sponsored by SDC, ARPA, and ESD" September 1 1965, Burroughs Corporation Records (CBI 90), Charles Babbage Institute, University of Minnesota, Minneapolis.

by wondering whether it was “possible that users, led by the military, will surrender to these data base systems without a shot being fired in anger.”⁶⁶

SDC’s attempt to push the data base concept into civilian discourse worked well. The term carried some specific associations, based on the particular characteristics of firms like SDC and of military command and control projects. One of these associations was with the idea of real-time operation – the data base would be constantly and, if possible, automatically updated with current information gathered from a number of different sources. It was also assumed that, as in SAGE, a data base could be “interrogated” in real-time by its users, answering questions interactively within seconds. In addition, the data base would be shared among many different programs, each one using only a subset of the overall information contained within it.

SDC had used its data base symposia to showcase its own on-line systems developed with military money. SDC’s most ambitious attempt to commercialize data base technology came with a system called CDMS (the Commercial Data Management System), a derivative of an earlier system called TDMS (Time-shared Data Management System) developed under contract from ARPA (Advanced Research Projects Agency) and given trial usage at military installations. These systems were intended to allow non-programmers to create data base structures, load data into them and then issue queries and retrieve their results on-line. Attempts to sell the TDMS computer program failed because it was expensive, needed a powerful computer all to itself, and could run only on SDC’s own custom-developed operating system. Attempts to rent use of CDMS through terminals connected to centralized computers were equally unsuccessful.⁶⁷

THE DATA BASE MANAGEMENT SYSTEM AND THE DBTG

The technological innovation represented by systems such as IDS was paralleled by conceptual developments. Until about 1968, the highfalutin data base concept remained fairly distinct from the practical world of tape based file management systems, report generators, and ever more disk-oriented packages such as IDS and IMS. The data base was much discussed but little realized. It was supposed to be used interactively on-line, could be used by non-specialists and was closely associated with the idea of a single huge reservoir of corporate information.⁶⁸ While SDC tried to sell this kind of tool there were few customers. In contrast file maintenance and report generation systems, and their more complex descendents such as IDS and IMS were used primarily by programmers to reduce development and maintenance costs for routine data processing applications.

Combining the *data base* and the *file management system* created the Data Base Management System. The DBMS idea was shaped and promoted through the work of a body called the Data Base Task Group (DBTG), an ad-hoc committee of the computer industry group CODASYL (Committee On Data SYstems Languages). CODASYL’s focus was the creation of data processing standards, and it is best known for its work designing and maintaining the COBOL programming language used for most business application programming from the late 1960s to the early 1990s. Its creation was prompted by

⁶⁶ R. V. Head, "Data Base Symposium" *Datamation*, vol. 11, no. 11, November 1965, pp. 41, page 41.

⁶⁷ C. Baum, *The System Builders: The Story of SDC*. Santa Monica: System Development Corporation, 1981, pages 116-121. A. H. Vorhaus, "TDMS: A New Approach to Data Management" *Systems & Procedures Journal*, vol. 18, no. 4, July-August 1967, pp. 32-35. Steig, "File Management Systems Revisited".

⁶⁸ See Haigh, "Inventing Information Systems".

the realization within CODASYL that COBOL, while doing a great deal to standardize data storage on tape systems and to separate record definitions from program logic, was entirely inadequate when faced with the challenge of random access, disk based storage. On its formation in October 1965 the DBTG had originally been called the List Processing Task Force (its name was changed only in 1967).⁶⁹ The group's founder and initial chair was W. G. Simmons of US Steel. It moved slowly for its first few years – an official history included in a later report noted that “because the membership of the group changed constantly a major amount of the DBTG's efforts was directed toward listening to and studying the views of as many persons as possible... [this] affected the progress rate of the group.”⁷⁰

The DBTG was dominated by the same manufacturers who were adding features to their file management systems and had begun to promote them as supporting, or even being, Management Information Systems.⁷¹ Its members were drawn from computer vendors, universities, consulting companies and a few large companies making heavy use of computers in their own business operations. Charles Bachman, the creator of IDS, was an early member of the committee and promoted the ideas he developed for IDS as the basis for its work.

As its name suggests, the DBMS was intended to be a new kind of product, extending the capabilities of existing file management packages to support the kind of advanced, on-line, interactive capabilities and huge integrated data stores associated with the data base concept. The purpose of the DBTG was to define the capabilities of these new systems, and to develop new standards for them.

In early 1968 the group presented a draft proposal, including a summary of Bachman's IDS, to the COBOL Language Subcommittee. In response the subcommittee approved the resolution that “COBOL needs the Data Base concept.” By January 1969 Appollon Metaxides of Bell Labs had taken over as chair, formalizing membership of the committee and focusing its work on the production of functional and language specifications for the COBOL's new capabilities.⁷² An initial public report specifying a Data Description Language and Data Manipulation Language was completed in October 1969 and published shortly afterwards as a draft for public comment.⁷³ 179 formal responses were received.⁷⁴ Its

⁶⁹ The phrase “data base management system” was used at least once before the renaming of the DBTG, to describe IBM's forthcoming Generalized Information System (GIS) J. H. Bryant and P. Semple, “GIS and File Management” in Proceedings of the 21st National Conference, Association for Computing Machinery, Ed., New York: ACM, 1966, pp. 97-107. List processing seems in retrospect and odd choice, but it may have gained cache through its association with work in artificial intelligence.

⁷⁰ CODASYL Data Description Language Committee, CODASYL Data Description Language: Journal of Development, June 1973. Washington, DC: U.S. Govt. Print. Off., 1974, 1.6.

⁷¹ W. G. Waites, “MIS or IMS?” Journal of Systems Management, vol. 22, no. 1, January 1971, pp. 32-34.

⁷² CODASYL Data Description Language Committee, CODASYL Data Description Language, 1.7.

⁷³ Anonymous, “CODASYL Data Management Report in Print” Data Base, vol. 1, no. 4, Winter 1969, pp. 4. The report was widely circulated, and large excerpts were published as CODASYL Data Base Task Group, “Data Base Task Group Report to the CODASYL Programming Language Committee” Data Base, vol. 2, no. 2 1970, pp. 11-18.

⁷⁴ CODASYL Data Description Language Committee, CODASYL Data Description Language, 1.10.

final recommendations were published in 1971 and endorsed by its parent group within CODASYL (the Programming Languages Committee).⁷⁵

Another CODASYL group, the Systems Committee, was chaired by William Olle of RCA who did important work in promoting the DBMS concept.⁷⁶ The Systems Committee worked on examining the capabilities of existing “generalized data base management systems,” issuing a hefty interim report in 1969.⁷⁷ The committee surveyed the strengths and weakness of existing systems such as GID, IDS, Mark IV, and TDMS, and began the attempt to identify a full list of desirable characteristics. It already included the concept of “data structure class” (equivalent to the later “data model”) which it used to characterize IDS as “hierarchical” and IMS as “network.” Despite lobbying by firms such as General Electric to get their own systems adopted as the basis for a new standard, the group decided that no single existing system came close to providing the range of features required. A second report, issued in 1971, added additional material, coverage of new systems, and analysis against the DBTG’s proposal.⁷⁸ These reports did a great deal to popularize the term “data base management system” as a category for products such as IMS, GIS and IDS which had not previously had a standard label.

The DBTG’s specific proposals were controversial at the time, and several CODASYL members opposed them (including mainframe suppliers IBM, RCA and Burroughs).⁷⁹ The IBM user groups SHARE and GUIDE had been conducting a joint data base specification project in opposition to CODASYL, beginning around 1969 with the creation of a wishlist for future systems.⁸⁰ The group was active until at least 1973, and presumably influence the SHARE’s board vote to oppose the adoption of the CODASYL proposals as industry standards.⁸¹ No complete implementation of the DBTG

⁷⁵ CODASYL Data Base Task Group, CODASYL Data Base Task Group: April 1971 Report. New York: Association for Computing Machinery, 1971.

⁷⁶ For example with T. W. Olle, "Recent CODASYL Reports on Data Base Management" in Data Base Systems, R. Rustin, Ed., Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1972, pp. 175-184 and a series of other articles as well as participation in a number of discussion panels such as V. C. Hare, Jr, "A Special Report on the SIGBDP Forum: 'The New Data Base Task Group Report'" Data Base, vol. 3, no. 3, Special Issue 1971, pp. 1. Olle tells the story of his era in T. W. Olle, "Nineteen Sixties History of Data Base Management " in IFIP International Federation fo Information Processing, Volume 215, History of Computing and Education 2 (HCE2), J. Impagliazzo, Ed., Boston: Springer, 2006, pp. 67-75.

⁷⁷ CODASYL Systems Committee, Survey of Generalized Data Base Management Systems, May 1969. This report was until recently hard to obtain outside the archival holdings of the Charles Babbage Institute, but has now been added along with other CODASYL materials to the ACM Digital Library.

⁷⁸ CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems.

⁷⁹ Hare, "A Special Report on the SIGBDP Forum: 'The New Data Base Task Group Report'".

⁸⁰ GUIDE/SHARE Data Base Requirements Project, "Resolution 69-001-00, November 4, 1969" in Proceedings of SHARE 34, March 2-6 1970: SHARE, Inc., 1970, pp. 697-712.

⁸¹ See material in the proceedings of SHARE 38 and 40 (March 1973). A good summary of the work of this group through 1972 is given in R. G. Canning, "The Debate on Data Base Management" EDP Analyzer, vol. 10, no. 3, March 1972, pp. 1-16. I am not sure whether the group eventually produced actual specifications, but its work does not appear to have had much influence on product developments.

specification was ever produced, and while some successful systems of the 1970s claimed to be heavily influenced by CODASYL many others did not. So CODASYL's project on DBMS languages were not as successful as its work on COBOL in setting industry standards. But in other ways its work proved very influential.

The work of the DBTG provided both a broad conceptual outline for the data base management system, and detailed draft specifications for two specific parts of the over-all system (a data definition language for defining the data base structure, and a separate data manipulation language for accessing the data from within COBOL). It also outlined a way of giving individual programs access to selective or simplified versions of the full data base.

The DBTG provided a new vocabulary for the field. It standardized terms such as "record" and "set" and "data base" and added some new ones, including "schema" (which remains ubiquitous today) to describe the logical format of data within the data base, and "sub-schema". A sub-schema (similar to what would be called a view in today's relational systems) allowed different users and applications to see only a portion of the overall database, allowing selective access to records and potentially shielding the application from changes in the underlying schema – a property referred to as "data independence."

The DBTG also separated the Data Manipulation Language (DML) used by programmers to add, delete, update and retrieve particular records from the Data Definition Language (DDL) used by the data base administrator to define the logical structure of the data base itself. That distinction remains a fundamental one in the data base field to this day. By May, 1969 CODASYL had realized that the DDL was not COBOL specific, and could be used to define data structures for use with other languages such as FORTRAN or PL/I (which IBM claimed would soon replace both COBOL and FORTRAN).⁸² While the DDL was to be a new and universally applicable language, the DML took the form of a programming language-specific set of additions seamlessly integrated into that language's existing instructions. This realization led CODASYL to split the work of the DBTG in two following approval of its 1971 report. The Data Manipulation Language Task Group, later renamed the Data Base Language Task Group, was responsible for reworking the proposed additions to COBOL into a form suitable for publication in CODASYL's "Journal of Development" as an official standard. This occurred in 1973.⁸³ The second group, the CODASYL Data Description Language Committee, was established as a standing committee to publish and enhance the standard DDL.⁸⁴ But these efforts to standardize a data definition language (DDL) failed to set a marketplace standard, in part because of the unwillingness IBM to

⁸² CODASYL Data Description Language Committee, CODASYL Data Description Language dates this decision to the Tenth Anniversary Meeting of CODASYL. It is reflected in the DBTG report issued later that year.

⁸³ The committee's work on the DML first appeared as CODASYL Database Language Task Group, CODASYL Cobol Database Facility Proposal. Ottawa: Dept. of Supply and Services, Government of Canada, Technical Services Branch, 1973 and was subsequently issued in the COBOL Journal of Development. Work on these standards continued into the 1980s, first through a new committee set up within CODASYL, and later at ANSI. This included a FORTRAN DML, to complement the COBOL DML in the earlier reports. CODASYL FORTRAN Data Base Manipulation Language Committee, CODASYL FORTRAN Data Base Facility, Journal of Development 110-GP-2. Ottawa: Dept. of Supply and Services, 1977.

⁸⁴ The DDL was published as CODASYL Data Description Language Committee, CODASYL Data Description Language.

commit to the network concepts inherent in the CODASYL model while its own flagship IMS product retained a hierarchical approach.⁸⁵

Its final contribution was to insist that a standard DBMS allow more complex linkages to be established between different files (or, as they were now to be called, record types) within the same data base. The DBMS was intended to make these relationships (or, as the DBTG called them, “sets”) as explicit and enforceable as previous file management systems had made the specification of fields within an individual file. Because most of the logic to maintain these relationships had previously been hidden within individual programs, placing relationships inside the DBMS along with the data itself ensured that all application programs and user requests would have access to them. The DBTG also endorsed Bachman’s idea of allowing complex networks of relationships between record types over the more restrictive hierarchical approach used by systems such as IMS.

This conceptual framework for the DBMS ultimately proved more influential than the DBTG’s detailed proposals. When the CODASYL work is mentioned at all today, it is usually for the propagation of the network data model. Since IDS, a commercial product, used this model prior to the DBTG’s establishment this would seem a rather limited contribution to history. In fact, the DBTG appears to have created, or at the very least to have publicly defined for the first time, the very idea of the data base management system as we know it today.

Though most of the characteristics that the DBTG specified for a DBMS had already been demonstrated by at least one file management or data base system, it insisted that future systems must provide all of them. A DBMS was expected to provide the efficient operational access for application programs and networked record-linking features that existing systems such as IDS specialized in. However, it was also expected to allow non-programmers to use a simple, specially tailored interface to query and update the data base directly – the province of systems such as Mark IV. Likewise a DBMS was expected to support interactive on-line usage, previously offered only by specialized systems such as SDC’s TDMS, and batch operation with equal felicity.⁸⁶ In practice most commercial systems of the 1970s failed to provide strong coverage across this wide spectrum of capabilities, but over time (and with the addition of external transaction processing systems such as CICS) they evolved toward the goal.

CONCLUSIONS

In 1973, Charles W Bachman was awarded the Association for Computing Machinery’s Turing Medal – the most prestigious award in computer science. The citation singled out his creation of the pioneering IDS system and his work on the DBTG to incorporate these ideas into its specifications. This award was in itself an important event, representing a new level of acceptance among computer science researchers of data base problems as intellectually respectable subjects of inquiry alongside better established areas such as numerical analysis, compiler theory and the theory of algorithms. The event is better

⁸⁵ See Performance Development Corporation, "An Interview With Charles W Bachman (Part II), Data Base Newsletter, Volume 8, No. 5" September 1980, Charles W. Bachman Papers (CBI 125), Charles Babbage Institute, University of Minnesota, Minneapolis. However, most of the advanced systems then under development were influenced to a more or less profound extent by ideas in the CODASYL reports – for a good summary of the most advanced commercial systems of the mid-1970s see R. L. Flynn, "A Brief History of Data Base Management" *Datamation*, vol. 20, no. 8, August 1974, pp. 71-7; Fry and Sibley, "Evolution of Data-Base Management Systems".

⁸⁶ CODASYL Systems Committee, Feature Analysis of Generalized Data Base Management Systems.

remembered, however, for Bachman's speech.⁸⁷ Entitled "The Programmer As Navigator," it developed the idea that the shift to DBMS technology represented something akin to the Copernican revolution – in that the work of programmers would now revolve around the data base rather than the hardware of the computer. Though this prophecy took several decades to come true, knowledge of data base systems has now become a fundamental requirement for virtually all administrative applications programming, systems analysis and advanced web design work.

The acceptance of the DBTG concept of a data base management system thus implied a new and more concrete vision of what a data base was – basically a body of electronic data that could be managed by a data base management system. Despite early hopes that the data base could be the heart of a system including all corporate information, it proved adept at handling only a small subset of this material. The data base, as realized through an extension of existing file processing tools, embodied the highly structured, administrative transaction-oriented view of information held by data processing staff and computer vendors.

The narrowing of the data base concept, and its close association with the DBMS, also represented a shift away from the idea, implicit in much earlier discussion of information retrieval, that all important information was scientific or at least was amenable to the same retrieval techniques as scientific information. The data base concepts pioneered by elaborate, military systems of the 1960s such as SDC's TDMS – on-line access, flexibly structured data, interactive definition of data formats by users, played little part in the leading commercial systems of the 1970s. Neither was there a significant commercial market for products based on these technologies. Though the industrial research budgets of leading corporations might have paid for subscriptions to the newly available on-line scientific data bases of the 1970s the managers and computing departments of the same companies had little interest in using these technologies to manage their own information.⁸⁸

The DBMS concept advanced by CODASYL proved far more important and longer lasting than the particular methods for its realization put forward by the DBTG. Despite their remarkable ubiquity, DBMSs based on the relational model continued to incorporate the same assumptions about information as earlier file management systems. As a result, the DBMS was very well suited to the bureaucratic records for things such as payroll administration, because each record included the same pieces of data (years of service, SSN, hourly rate, overtime status and so on). It made it very simple and efficient to update information, and so is well suited to administrative systems where records are constantly updated. On the other hand, it was entirely useless for representing and searching less rigidly formatted data, such as full-text records, correspondence, or even scientific abstracts. Only with the rise of the World Wide Web in the mid-1990s did widespread attention turn back to the indexing and management of huge amounts of natural language information.

ACKNOWLEDGMENTS

Thanks to Mary Ellen Bowden for encouraging me to begin this research, to Boyd Rayward for his close attention to my earlier work on this topic, to Rick Snodgrass and ACM SIGMOD for funding my oral history interview with Charles W. Bachman and to Burt Grad and the Computer History Museum for

⁸⁷ C. W. Bachman, "The Programmer as Navigator" *Communications of the ACM*, vol. 16, no. 11, November 1973, pp. 653-658.

⁸⁸ The commercial emergence of on-line information services, discussed in great detail in C. P. Bourne and T. B. Hahn, *A History of Online Information Services: 1963-1976*. Cambridge, MA: MIT Press, 2003, appears to have taken place in almost complete isolation from work on DBMS packages.

supporting my oral history interview with Robert L. Patrick. Some material here is adapted from W. Boyd Rayward and Mary Ellen Bowden, eds., *The History and Heritage of Scientific and Technological Information Systems: Proceedings of the 2002 Conference* (Medford, New Jersey: Published for the American Society for Information Science and Technology and the Chemical Heritage Foundation by Information Today, Inc., 2004.)