A11103 090169

COMPUTER SCIENCE & TECHNOLOGY:

DATA BASE DIRECTIONS—
The Conversion Problem

NBS Special Publication 500-64
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

)0-64

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress on March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to the public and private sectors to address national needs and to solve national problems; conducts research in engineering and applied science in support of these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the ultimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following centers:

Programming Science and Technology — Computer Systems Engineering.

[1]Headquarters and Laboratories at Gaithersburg, MD, unless otherwise noted; mailing address Washington, DC 20234.
[2]Some divisions within the center are located at Boulder, CO 80303.

National Bureau of Standards
Library, E-01 Admin. Bldg.
OCT 29 1980
not circ - circ
QC100
.U57
NO 500-64
1980
C.2

COMPUTER SCIENCE & TECHNOLOGY:
DATA BASE DIRECTIONS—
The Conversion Problem

Proceedings of the Workshop of the
National Bureau of Standards and the
Association for Computing Machinery,
held at Fort Lauderdale, Florida,
November 1 - 3, 1977

John L. Berg, Editor:

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

Daniel B. Magraw, General Chairperson

Working Panel Chairpersons:

Milt Bryce, James H. Burrows,
James P. Fry, Richard L. Nolan

Sponsored by:

National Bureau of Standards

Association for Computing Machinery

Special publication

U.S. DEPARTMENT OF COMMERCE, Philip M. Klutznick, Secretary

Luther H. Hodges, Jr., Deputy Secretary
Jordan J. Baruch, Assistant Secretary for Productivity, Technology and Innovation

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued September 1980

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in this series should complete and return the form at the end of this publicaton.

National Bureau of Standards Special Publication 500-64
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-64, 178 pages (Sept. 1980)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 80-600129

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1980

For sale by the Superintendent of Documents, U.S. Government Printing Office
Washington, D.C. 20402 - Price $5.50

TABLE OF CONTENTS

Page

1. INTRODUCTION 3

1.1 THE FIRST DATA BASE DIRECTIONS WORKSHOP 3

1.2 PLANNING FOR SECOND CONFERENCE 4

1.3 DATA BASE DIRECTIONS II 4

1.4 CONCLUSION 5

2. EVOLUTION IN COMPUTER SYSTEMS 7

2.1 QUESTIONS 7

2.2 HARDWARE CHANGES 9

2.3 SOFTWARE CHANGES 10

2.4 EVOLUTIONARY APPLICATION DEVELOPMENT 11

2.5 MIGRATION TO A NEW DBMS 13

3. ESTABLISHING MANAGEMENT OBJECTIVES 19

3.1 OVERVIEW 20

3.2 CONVERSION TO A DATA BASE ENVIRONMENT 24

3.2.1 Impact On the Application Portfolio. 25
3.2.2 Impact On the EDP Organization. 29
3.2.3 Impact On Planning and Control Systems. ... 35
3.2.4 Impact of Conversion On User Awareness. ... 38

3.3 MINIMIZING THE IMPACT OF FUTURE CONVERSIONS .. 39

3.3.1 Institutionalization of the DBA Function. . 39
3.3.2 DBMS Independent Data Base Design. 40
3.3.3 Insulate Programs From the DBMS. 40

3.4 CONVERSION FROM ONE DBMS TO ANOTHER DBMS 41

3.4.1 Reasons for Conversion. 42
3.4.2 Economic Considerations. 43
3.4.3 Conversion Activities and Their Impact. ... 44
3.4.4 Developing a Conversion Strategy. 48

3.5 SUMMARY 50

4. ACTUAL CONVERSION EXPERIENCES 53

4.1 INTRODUCTION 54

4.2 PERSPECTIVES 55

4.3 FINDINGS 57

4.3.1 Industrial/Governmental Practices. 58
4.3.2 The First DBMS Installation. 60
4.3.3 Desired Standards. 61
4.3.4 Desired Technology. 61

4.4 TOOLS TO AID IN THE CONVERSION PROCESS 62

4.4.1 Introduction. 62
4.4.2 Changing From Non-DBMS To DBMS. 62
4.4.3 Changing From One DBMS To Another. 64
4.4.4 Changing Hardware Environment. 66
4.4.5 Centralized Non-DBMS--distributed DBMS. ... 66
4.4.6 Centralized DBMS--distributed DBMS. 67

4.5 GUIDELINES FOR YOUR FUTURE CONVERSIONS 67

4.5.1 General Guidelines. 68
4.5.2 Important Considerations. 68
4.5.3 Tight Control. 69
4.5.4 Precise Planning/pre-planning. 69
4.5.5 Important Actions. 70

4.6 REPRISE 73

4.7 ANNEX: CONVERSION EXPERIENCES 73

4.7.1 Conversion: File To DBMS. 73
4.7.2 Conversion: Manual Environment To DBMS. .. 79
4.7.3 Conversion: Batch File System To a DBMS. . 84
4.7.4 Conversion: DBMS-1 To DBMS-2. 87

5. STANDARDS 93

5.1 INTRODUCTION 93

5.1.1 Objectives. 93
5.1.2 What Is a Standard? 94
5.1.3 Background. 94

5.2 POTENTIAL BENEFITS THROUGH STANDARDIZATION ... 97

5.3 SOFTWARE COMPONENTS IN CONVERSION PROCESS 98

5.3.1 Scenario 1. 98
5.3.2 Scenerio 2. 99
5.3.3 Scenario 3. 100
5.3.4 Scenario 4. 100
5.3.5 Miscellaneous Standards Necessary. 100
5.3.6 Non-software Components Necessary. 100

5.4 RECOMMENDATIONS 101

5.4.1 The Development of a Standard DBMS. 101
5.4.2 Generalized Dictionary/directory System. . 101

5.5 CONCLUSION 103

5.6 REFERENCES 103

6. CONVERSION TECHNOLOGY--AN ASSESSMENT 105

6.1 INTRODUCTION 106

6.1.1 The Scope of the Conversion Problem. 106
6.1.2 Components of the Conversion Process. 107

6.2 CONVERSION TECHNOLOGY 109

6.2.1 Data Conversion Technology. 110
6.2.2 Application Program Conversion. 120
6.2.3 Prototype Conversion Systems Analysis. ... 129

6.3 OTHER FACTORS AFFECTING CONVERSION 138

6.3.1 Lessening the Conversion Effort. 138
6.3.2 Future Technologies/Standards Impact. 145

BIBLIOGRAPHY 150

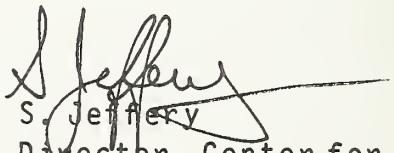
7. PARTICIPANTS 161

PREFACE

In 1972 the National Bureau of Standards (NBS) and the Association for Computing Machinery (ACM) initiated a series of workshops and conferences which they jointly sponsored and which treated issues such as computer security, privacy, and data base systems. The three-day Workshop, DATA BASE DIRECTIONS--The Conversion Problem, reported herein continues that series. This Workshop was held in Fort Lauderdale, Florida, on November 1-3, 1977, and is the second in the DATA BASE DIRECTIONS series. The first, DATA BASE DIRECTIONS--The Next Steps, received wide circulation and, in addition to publication by NBS, was published by ACM's Special Interest Group on Management of Data and Special Interest Group on Business Data Processing, the British Computer Society in Europe, and excerpted by IEEE and Auerbach.

The purpose of the latest Workshop was to bring together leading users, managers, designers, implementors, and researchers in database systems and conversion technology in order to provide useful information for managers on the possible assistance database management systems may give during a conversion resulting from an externally imposed system change.

We gratefully acknowledge the assistance of all those who made the Workshop's results possible.



S. Jeffery
Director, Center for Programming
Science and Technology
Institute for Computer Sciences and
Technology

A MANAGEMENT OVERVIEW

To a manager, conversion answers the question, "How do I preserve my investment in existing data and programs in the face of inevitable changes?" Selection of conversion as a solution depends directly on issues of cost, feasibility, and risk. Since change is inevitable, prudent managers must consider preparations that ease inevitable conversions. How does a manager choose a course of action?

When Mayford Roark, Executive Director of Systems for the Ford Motor Company, and Keynoter of the Workshop, sought an analogue for examining DBMS and the conversion problem, he aptly selected the idea of "mapping a jungle." Reporting on his experience, he noted that 90% of his computers were changed within three to five years and major software changes from the vendor occurred somewhere between five and ten years after acquisition.

These forced changes coupled with the organization's changing requirements led Roark to a basic point: "evolutionary change is the natural state of computer systems." In short, the ADP manager's continual task is to "manage change."

Roark's managers experienced the classic benefits of DBMS: quicker response to changing requirements, easier new application development, and new capabilities not possible in the earlier systems, but Roark summarized his conversion experience within a DSBMS environment in this way:

- hardware changes--having a DBMS was a moderate to major burden.
- software changes--dependent on the circumstances, having a DBMS ranged from negligible impact to a major burden.
- evolutionary application change--having a DBMS was a moderate boon. It proved effective, but very expensive.

Considering even the conversion to a DBMS, Roark scores this process a moderate burden because of the risks and costs associated with DBMS.

He emphasized this point by describing the major DBMS need as "easy-to-use, easy-to-apply, and inexpensive approaches for upgrading" two decades of computer files to a data base environment.

Given this charge, how did the workshop respond?

Consideration of the conversion to a DBMS led to several specific caveats intended to control the risk inherent in such a step.

- Though conversion to a DBMS requires careful preplanning and may not be appropriate for every application, managers should consider data base technology an inevitable development thrusting itself on future data processing installations. A manager will have no choice but to face this decision eventually.

- The first DBMS application can make or break the success of the conversion. The new system's users must have a receptive disposition which results only from careful preparation, preplanning, and the application of basic management skills. The initial application plays an important tutorial role for everyone in the organization including such subtle lessons as:

  --whether management is truly committed to the new system by supporting it with adequate resources, planning for the continuous support of the system, and applying the necessary managerial cross-department discipline.

  --whether the installation technical staff truly appreciates user needs, can adjust to user changes, and has the necessary skills and backing to carry-off the task.

  --whether any of the DBMS conversion proponents have accurate estimates of the costs, the proper tools to use, and a feasible conversion plan expressed in terms that satisfy <u>all</u> risk sharers.

- While the staff must know the new technology, they must not conclude that the new technology relieves them of the old project management controls that all new systems require. Tight planning, management control, cost monitoring, contingency approaches, user review, and step-wise justifications must be used.

- No "final conversion" exists--planning for the next one begins now! Prepare your system to permit evolutionary change to enhanced technology--including improvements to DBMS.

How will future technology help managers? Hardware development, particularly the proliferation of mini- and micro-computers, networks, and large mass storage, will increase the need for generalized conversion tools. On the other hand, dropping hardware costs will make conversion increasingly more acceptable. Additionally, special DBMS machines which promote logical level interfacing will simplify the conversion process. Major advances in improved data independence through software design will also simplify but never eliminate the conversion problem. Of special concern to managers: user demand for several data models will continue.

In the next five years, managers can expect to see more operational generalized conversion tools but certainly not full automation of the process. Significantly, standards design and acceptance by vendors will plan a major role in the success of generalized conversion tools. Commercially available tools for data base conversion seem likely in ten years but the conversion of application programs is not likely to have a generalized solution in the next five years.

Standards address several manager needs in the conversion process. A standard DBMS would considerably ease the future conversions involving a DBMS. A standard data dictionary/directory would facilitate all conversions. This latter point emphasizes that the data dictionary/directory can stand apart from data base systems and, therefore, can assist the conversion to a first DBMS.

A standard data interchange format would ease considerably the loading and unloading of data and thus facilitate the development of generalized conversion tools. Manufacturer acceptance of the standard format would permit their development of convertors from the standard form to their system, a boon to managers either forced to or desirous of considering a different system.

Similarly, standardization of the terminology used in data base technology, convergence of existing DBMS to using common functions, and use by DBMS of a micro-language or standard set of atomic functions would assist managers in dealing with conversion from DBMS to DBMS.

In summary: in the next five to ten years managers must depend on existing good management practices rather than wait for automated conversion tools. Standards, as a management exerted discipline, will facilitate conversions but users can expect reluctant acceptance of standards.

DATA BASE DIRECTIONS--
The Conversion Problem

John L. Berg, Editor

ABSTRACT

What information can help a manager assess the impact a conversion will have on a data base system, and of what aid will a data base system be during a conversion? At a workshop on the data base conversion problem held in November 1977 under the sponsorship of the National Bureau of Standards and the Association for Computing Machinery, approximately seventy-five participants provided the decision makers with useful data.

Patterned after the earlier Data Base Directions workshop, this workshop, Data Base Directions--the conversion problem, explores data base conversion from four perspectives: management, previous experience, standards, and system technology. Each perspective was covered by a workshop panel that produced a report included here.

The management panel gave specific direction on such topics as planning for data base conversions, impacts on the EDP organization and applications, and minimizing the impact of the present and future conversions. The conversion experience panel drew upon ten conversion experiences to compile their report and prepared specific checklists of "do's and don'ts" for managers. The standards panel provided comments on standards needed to support or facilitate conversions and the system technology panel reports comprehensively on the systems and tools needed--with strong recommendations on future research.

Key words: Conversion; Data Base; Data Description; Data Dictionary; Data Directory; DBMS; Languages; Data Manipulation; Query.

1. INTRODUCTION

Daniel B. Magraw

GENERAL CHAIRMAN

Biographical Sketch

Daniel B. Magraw is Assistant Commissioner, Department of Administration, State of Minnesota. For nearly ten years he has been responsible for all aspects of the State of Minnesota information systems activities. His more than thirty years' experience in systems divides almost equally between the private and public sectors.

A frequent contributor to professional activities, he was one of the founders and is a past president of the National Association for State Information Systems. He taught courses in Systems for 22 years in the University of Minnesota Extension Division and he has been a frequent speaker on many matters relating to information systems and has been deeply involved with both Federal and state data security and privacy legislation.

He was keynote speaker at the 1975 Data Base Directions Conference.

1.1 THE FIRST DATA BASE DIRECTIONS WORKSHOP

In late October, 1975, a Workshop entitled "Data Base Directions: The Next Steps" was held in Fort Lauderdale, Florida. Resulting from a proposal brought to Seymour Jeffery at the National Bureau of Standards by Richard Canning and Jack Minker, the workshop was sponsored jointly by the Association for Computing Machinery and NBS. The product of the intensive two and a half day effort was a series of panel reports which, as subsequently edited, were issued under the title of the workshop as NBS Special Publication 451.

As early as December, 1975, suggestions were made to ACM and NBS concerning the desirability of one or more future conferences on the same general topic. These suggestions were based on the belief that data base systems will grow increasingly in importance and pervasiveness and were supported by perceptions, even prior to issuance of NBS SP 451, that the workshop had more than met expectations.

The report was issued in 1976; it was generally thought to be a valuable contribution to several audiences including top management, EDP management, data base managers, and the industry. It has had an unusually wide circulation for reports of this nature.

1.2 PLANNING FOR SECOND CONFERENCE

In February, 1977, ACM and NBS decided in favor of a second data base workshop and invited me to serve as General Chairman. An initial planning group was established consisting of Dick Canning and Jack Minker representing ACM, John Berg representing NBS, and the chairman. The planning group entitled the Conference "Data Base Directions II--The Conversion Problem."

Four working panel subjects were selected. Panel chairmen were recruited and became members of the planning group. Subject matter coverage for each panel was specified by the planning group. Each panel chairman then selected members of his working panel. In addition, the planning group specified that the format and procedure for the workshop should follow closely that of the first workshop. As in the 1975 workshop, attendance was by invitation only. Work was done by each panel prior to arriving at the workshop.

1.3 DATA BASE DIRECTIONS II

The workshop was held November 1-3, 1977, in Fort Lauderdale, Florida. There were approximately 75 in attendance. Mayford Roark, Ford Motor Company, gave an excellent keynote address in plenary session, reproduced herein. Final instructions were given as to objectives of the workshop, and all panels were in full operation by 10:30 a.m., November 1. From then until the closing plenary session, each of the four panels met separately with the ultimate purpose of developing a consensus among its members on which to base the panel report.

Each working panel chairman was expected to guide the group discussion and to assure preparation of a good rough draft report prior to the closing plenary session. Though each panel chairman organized his panel in a form best for his subject, each followed a general pattern. A recorder was selected from among each panel's members to maintain "minutes" in visual form on flip chart sheets. These were displayed so that the principal discussion and consensus points were visible.

One of the panels had done extensive drafting of report segments prior to the beginning of the workshop. Each of the other three accomplished the objective of rough draft preparation by developing detailed outlines. Portions of the outlines were assigned to individual panel members or to two person teams for draft preparation. When time permitted, drafts were reviewed by others on the same panel prior to their incorporation into the final draft.

Following the completion of the workshop, the drafts were put into "final" form and circulated to panel members for final comment prior to submission to the proceedings editor.

Communication among the four panels was maintained mainly through the panel chairmen and members of the planning group who circulated among the panels. At the closing plenary session, each working panel chairman presented a twenty minute summary of his panel's findings and responded to questions or comments from the floor.

1.4 CONCLUSION

The mix of participants with their differing perspectives, experiences, and technical expertise provides a base of knowledge in data base systems that may be unparalleled. The output from that group should be of material value, especially to those decision-makers across the country carrying responsibility for data base futures. It is hoped that this publication can have an even greater impact than did Data Base Directions I because it is based on two more years of extensive experience and because it attempted to be more direct and pointed in its advice to the decision- makers. The real value of Data Base Directions II, however, will be directly proportional to the assistance that this document provides to the several classes of users.

2. EVOLUTION IN COMPUTER SYSTEMS

Mayford L. Roark

KEYNOTER

Biographical Sketch

Mayford L. Roark is Executive Director of Systems for the Ford Motor Company in Dearborn, Michigan. He has been in charge of the corporate systems function at Ford since 1965, as Assistant Controller and later as Director of the Systems Office before assuming his present position in 1973. He joined the Ford Division of Ford Motor Company in 1952 as Senior Financial Analyst, and managed various financial departments at Ford from 1955-1965.

Mr. Roark was a Budget Examiner at the U.S. Bureau of the Budget from 1947 to 1952. Previously he was with the U.S. Weather Bureau and the Colorado Department of Revenue.

He studied at the University of Colorado, receiving the B.A. "Magna Cum Laude" degree (Economics), and the M.S. (Public Administration). He is a member of Phi Beta Kappa.

2.1 QUESTIONS

When I was asked to address this group, with its theme of "The Conversion Problem," I felt some puzzlement about the thrust of the meeting. Was there a concern that data base systems might be something of a special burden for the organization faced with a conversion problem? Or, rather, was there the hope that the data base system might be something like "Seven League Boots," for making otherwise tough conversions into happy and speedy journeys? Or, was there a lingering horror that the data base systems themselves might turn out to be conversion nightmares as improved DBMS resources become available?

As the working outlines began to arrive through the mail, it became clear that all of these questions were concerns. As Jim Fry's statement put it, "The basic problem we are addressing is the need to migrate data and applications due to the introduction of new, or change in the current requirements, hardware, and/or software systems." In short, it appears that your intent is to map a jungle.

I won't try to preempt the work of the individual panels by offering any detailed map of my own. Rather, as a frequent traveler through this jungle, my traveler's notes may be of some help to your individual survey parties. My comments will take the form of generalizations and impressions. As you survey a topic in depth, you may well conclude that some of these were inaccurate. This has certainly been the usual result in geographic history, as detailed surveys refine the rough findings of early travelers.

As an initial generalization, let me categorize conversion problems into four families of change:

. Hardware Changes,

. Software Changes,

. Evolutionary Application Development,

. Migration to a New DBMS.

I should like to discuss each of these families of conversion problems, and to relate my own impressions as to their frequency of occurrence and their significance with respect to data base management systems. In an effort to be as specific as possible, I will try to rate the impact of a DBMS in each case against a "BB scale"--that's shorthand for "Boon or Burden." If a DBMS, in my view, is a major boon or help for a given class of conversion problem, I shall score it as plus 2. If it is a moderate boon, that will be plus 1. If DBMS is more likely to be a burden than a boon, I shall score it as minus 1 or minus 2. Now let me proceed to a discussion of these families of conversion problems.

2.2 HARDWARE CHANGES

One of my chores at Ford is to undertake, with the aid of others, a "prior review" of every computer hardware acquisition involving a change of processor or an added rental of more than $10,000 a year. I have been doing this for a dozen years now, during which time I have signed off on about 3,000 computer projects. If Ford is typical of the universe, I would guess that 90 percent of all data processing computers are likely to be replaced within 3 to 5 years from their acquisition. This would not hold for minicomputers used in dedicated, industrial control applications; these may remain in operation without significant change for a decade or more. Data processing computers do change fairly rapidly, however. For most of us, 3 to 5 years is long enough to saturate the capacity of whatever computer we have and to move on to something more powerful. We are also faced with a dazzling succession of new products every few years, always with substantial improvements in cost effectiveness. So, for most of us, I think the 3 to 5 year cycle is likely to continue for awhile.

The extent of conversion trauma from a hardware upgrade depends on the nature of the change. If the change involves shifting to another supplier, an event that occurred in something like 5 percent of our hardware changes, the conversion effort can be an extended affair requiring a major effort over a period of a year or more. In such cases, a DBMS is likely to be something of a burden. In all probability, the new equipment will require a shift to a different DBMS. In any event, the logic requiring conversion will be somewhat more complex and difficult to handle if a DBMS is present.

One of our divisions recently completed a conversion from Honeywell to Burroughs, a change made necessary because of reorganization unrelated to the system function. Some of the application programs to be converted had been developed in Honeywell's IDS environment; they were converted to Burroughs DMS II. IDS is a network system; DMS II is a hierarchical structure, so one might guess that we would have had problems. In fact, I am assured, both by our own people and by the Burroughs conversion team, that this transition went fairly smoothly. On the basis of that once-in-a-lifetime experience, I would have to score the DBMS in this case as minus 1. Perhaps it would have been minus 2 if the application programs involved had been larger and more complex.

Hardware conversion can also be a serious trauma when migrating to new products of an incumbent supplier. Suppliers do present us with new families of hardware from time to time that require structural changes in our application programs and supporting software. Hopefully, this kind of change will be less frequent in the future than it has been in the past. Even so, I suspect we are likely to be faced with something of this kind every 10 years or so. Already, one hears strange rumblings about the kinds of changes likely to unfold 2 or 3 years hence in a new product line called "The New Grad." So far, DBMS has not figured importantly in any of our conversions within the hardware products of a given supplier. On the basis of guesswork and a skeptical disposition, I would have to assume that the presence of a DBMS for a major hardware family transition would be similar to that where a change in supplier is involved, possibly minus 1 or minus 2 on the BB scoring scale.

The remaining hardware conversions, which represent the vast majority of all hardware upgrades, involve moving up within a compatible family of products offered by a single supplier. In these cases, DBMS would not be much of a factor, so we might score its presence as zero on the BB scale.

2.3 SOFTWARE CHANGES

Despite IBM's assurance that MVS is here to stay, our experience would suggest that we can look for "major" software upgrades or enhancements from any supplier every 5 or 10 years. At Ford, we are somewhat preoccupied at the moment with the MVS problem at our large data center in Dearborn. The completion of this conversion will require a major effort extending over a two-year period. There are three substantial groups of data base systems that will be affected by the conversion; these involve, respectively, IMS, TOTAL, and SYSTEM 2000. I have checked with each of the systems groups involved to see how they assess the impact of DBMS at this point, when we are about midway in the conversion effort. The managers working with IMS agree that its presence has added substantially to the difficulty and complexity of the conversion task--something close to twice as much effort as would have been involved with non-DBMS applications. The IMS conversion involves more than a new operating system, however. It requires the transition to a new DBMS called IMS-VS. One manager sees the new package as offering many attractive new features. Another manager sees no incremental benefits at all for his particular applications; but he has no choice, his present IMS software will not be supported under MVS.

The managers using TOTAL and SYSTEM 2000 see no special conversion problem at all in going to MVS.

This sampling of experience adds up to a very mixed bag. As the warnings say about some drugs, a major software conversion "may be hazardous to your health"--but again, it may not. I think we have to score the presence of a DBMS in such a situation with a range from 0 to minus 2 on the BB scale.

We do have numerous other software changes, of course, on an almost continuous basis--new releases to old operating systems, new software packages to handle new functions, and the like. There is nothing in our experience to indicate that DBMS is much a factor one way or another in these minor changes.

2.4 EVOLUTIONARY APPLICATION DEVELOPMENT

Now we move into the territory of what the systems function is all about, and what DBMS also is all about.

Before getting into the particulars about this family of change, we ought to be clear on one central point. "Evolutionary change" is the natural state of computer systems, just as it is for biological systems. Perhaps I should not push this analogy too far because there is one dramatic difference--evolutionary change works more rapidly in computer systems, where even 5 years can produce dramatic changes in structure, outputs, and even objectives.

During the past 5 years, the workload at our major computer centers at Ford has been growing at close to 20 percent annually. I might explain here that we attempt to measure workload in BIPS (Billions of Instructions Processed) for purposes of capacity planning. So, when I say that growth has been close to 20 percent a year, I mean that the BIPS have been growing at close to 20 percent annually.

As best as I can judge, about half of this growth results from new applications and about half from new adaptations of existing applications.

The new adaptations, which are often described by the rather condescending term "maintenance," include a lot of things. One is simply the effect of volume. If we sell more cars, other things equal, we will process more BIPS. Another is changing requirements. In our industry, every model year is, in a sense, a new game. The products may change, terminologies may change, and data requirements may

change. One of the biggest sources of changing requirements in the automotive industry is government regulation. The work of the regulators never ceases, nor does the growth in requirements for additional data elements in the burgeoning computer records that we must maintain as evidence of compliance with all sorts of directives from Washington.

Some of these changing requirements are massive things like OSHA regulations or like corporate fuel economy standards. Others seem almost trivial until they are examined in the context of required changes in computer files. The industry is currently being asked to adopt as an international standard a 16-character vehicle identification number. Our existing identification numbers, with 11 characters, turn up in more than 50 separate computer files in North America alone. The cost of converting these files and their related application programs to the new international standard is estimated at $3 million.

In all fairness, the government is not the only source of changing requirements, our engineers and product planners are pretty good changers themselves. Our service parts files include roughly twice as many parts today as 10 years ago, when our basic inventory control system was developed. Our organization people contribute more than a fair share of changing requirements. Every time there is a corporate reorganization, we find it necessary to go through a restructuring of the hundreds, or even thousands, of computer files and application programs that are affected by the realignment.

And even systems people are contributors to the evolutionary process, only we usually call the changes "efficiency improvements." Almost every systems activity has its own more or less continuous effort aimed at cleaning up programs that run inefficiently, that are hard to maintain, or that otherwise need overhaul.

To make another sweeping generalization, I would judge that each of our systems activities annually rewrites somewhere between 5 percent and 25 percent of its accumulated code.

What a fantastic opportunity for data base management systems! I would score the DBMS impact on this area of evolutionary application development as plus 2 and proceed to the next topic except for one problem. The DBMS benefits can be extremely difficult to realize in practice, and the price of the cure is sometimes worse than the pain of the ailment.

Many of our files most subject to change are huge affairs. Some of these files run into the billions of bytes of data. For systems of this size, processing costs may be counted in 6 or 7 figures annually. An added overhead burden in a range of 30 percent to 40 percent can amount to several hundreds of thousands of dollars in added annual cost. One of our activities last year made an analysis of overhead associated with one of its large data base systems and found that 70 percent of the instructions being executed resulted from DBMS overhead. This activity is in the process of restructuring its DBMS with the objective of reducing processing requirements by a full 40 percent.

Several years ago, another of our activities launched a large-scale data base system in which nearly half of the file requirements were required for pointers. Processing requirements in such a case can far exceed expectations based on any prior experience. This sort of overhead can mean a loss in processing productivity, something that has to be weighed against any benefits in programming productivity.

The best score I can give DBMS, therefore, for its impact on "evolutionary application development" is plus 1. It sometimes works well, but it can also be awfully expensive.

Somewhere in the deliberation of this "or related groups," there ought to be some consideration of what can be done to simplify DBMS technolgy. If this is not possible, perhaps there could be some guides or standards to protect the systems designer with a modest problem from unwittingly stumbling into a huge solution out of all proportion to his need. Sometime in the future, I would like to go through an exercise like this again and conclude, without any reservation, that DBMS is an unqualified boon for the evolving system regardless of its size and complexity. We are still in the very early stages of DBMS art.

2.5 MIGRATION TO A NEW DBMS

I have saved this family of conversion problems until last. In a sense, it overlaps all three of the categories I discussed earlier. Migration to a new DBMS may be prompted by a change of hardware; it may be forced by a change of software; or it may be undertaken with a view to realizing the benefits we discussed under "evolutionary applications development." Still, the migration to a new DBMS is a major event that deserves consideration in its own right, whether the migration is from a non-DBMS environment or is the rare change from one DBMS to another.

There is a logical inconsistency in assigning any rating at all to this family of conversion problems. It is like asking, "What impact does DBMS have upon itself?" Yet, at the risk of sounding completely illogical, I am going to assign a BB rating of minus 1 to this category of conversion problems on the ground that a DBMS is a high-risk undertaking entirely apart from whether it is related to changes in hardware, software, or applications.

Moving to a new DBMS is not unlike the process of getting married, it takes a lot of desire, commitment, sacrifice, and investment. It involves moving to a wholly new lifestyle. Once there, the return to the old lifestyle may be difficult or impossible without wrenching adjustment problems.

I have several times had the experience of encountering a spokesman for some other organization who tells me, "We've made the policy decision that all future development work in our organization will be based on DBMS." This makes me shudder. It is a little like saying, "We've decided that everyone should get married." I must add, however, that in every instance cross examination has established that, policy or no policy, the organization in question still does a lot of its computer business outside the DBMS environment. I expect this will be true of nearly all of us for many, many years to come, or at least until the DBMS technology can be simplified to the point where it no longer requires total desire, total commitment, total preparation, and heavy investment.

Not too long ago, we compiled a catalog of all the computer systems applications in use at our North American divisions and affiliates. The listing came to something like 50,000 application programs. Some of these resulted from major projects requiring scores of man-years to develop. Others were relatively simple. The full range of applications represents a wide spectrum of data needs and complexity. DBMS technology today does not really address this whole range of developmental requirements. Perhaps it never should. In any event, migration to a new DBMS system must be taken as one of life's climactic events to most systems people. We all need more guidance about when to undertake such a migration and how to stay out of trouble when we do.

These, then, are the major problems to be found in this jungle of systems conversions. To summarize, my BB scorings have suggested that a DBMS can constitute a moderate-to-heavy burden for major hardware conversions and major software conversions. The presence or availability of DBMS, however, can be a moderate-to-strong boon for evolutionary

application development. Finally, the migration to a DBMS system can be a long and tedious journey, especially where existing systems of great size and complexity are involved.

DBMS, in short, still offers attractive visions of a world in which systems might respond quickly to changing requirements, in which new creative applications might be easily spawned from existing data files, and where data bases,in the words of Dan Magraw at the earlier conference of two years ago, can "move the DBM's into the area of decision making."

These visions ought not to be taken lightly. In preparing for this meeting, I checked back with our managers who have been in the DBMS mode for 5 years or more. What benefits did they really get? Their consensus might be summarized as follows:

1. They, indeed, have been able to respond more quickly to changing requirements, although not always as easily as they once hoped.

2. New applications development has been easier. The managers see a productivity improvement factor in a range of 10 percent to 20 percent.

3. Most important, the managers believe they have capabilities that previously did not exist at all.

Let me expand on these capabilities for a moment. Those 50,000 computer programs I mentioned are a long-time accumulation in response to numerous problems and opportunities that were perceived by our division and staffs over a period of two decades.

Our typical division, which might correspond to a moderate-size company, has its own accumulation, usually a range of 1,500 to 3,000 programs. If the division is not yet in a DBMS environment, each of those programs comes with its own set of files. In the last two years, we have been greatly influenced by the concepts of "top-down design" and "structured programming." For systems of recent vintage, these approaches may have provided a logical structure that would give some accessibility to files. For older systems, with what our people call "spaghetti programs," accessibility is something else--fragmented and scattered files, with all the access keys hidden in "spaghetti code."

As functional entities for the purposes they were first created, these old programs may serve well. Even where we want to launch a new application that will need to draw on the data in these files, the problem is not overwhelming.

We can, and do, write programs to get at the needed data files, wherever they exist.

But suppose we do not want a new system. All we want is an in-depth analysis of a difficult problem on which 5 or 10 different files have something useful to say. This is a sort of problem that gives computer people a bad reputation as being slow-moving and unresponsive. It can be extremely difficult to extract data from 5 or 10 different sources, all with different maintenance cycles and data control procedures, and produce anything but a mess.

We do a lot of computer-based analysis at Ford because we have found that the data resources in our computer files can open up all sorts of insights and understanding that would otherwise be lost. I wish we could do even more, but this form of analysis can be very time consuming and frustrating in the non-DBMS environment. We have been working on one such exercise involving non-DBMS file sources for more than three months, all because of the relative inaccessibility of data. Last week we decided to skip a promising analysis altogether because it would have taken more than a month to extract and organize the needed data from a variety of source files.

So, when our managers talk about new capabilities from DBMS, they are talking about one of the computer's most important potentials--the power to carry analysis to entirely new levels of understanding.

The benefits cited by our managers are impressive testimonials. Why, then, has the data processing world been so slow in converting to DBMS? I have seen no studies that would provide an accurate measure as to the proportion of data processing oriented to DBMS. In the absence of any sure data, I am going to offer the opinion that the proportion is no greater than 10 percent to 20 percent.

I have never expected that all computer systems would, or should, move to DBMS. In the early Seventies, however, as we first began to realize the potential of this technology, most of us, even the conservatives I believe, would have expected that something approaching half of all computer files would acquire a DBMS format by the end of 1977. Even at Ford, where I believe the impact of DBMS has probably been greater than average, the progress has seemed slower than I would have expected. This painfully slow progress points up perhaps the greatest conversion problem of all--how can we move to a DBMS environment with existing systems?

Most of our DBMS user divisions made their first moves to data base at least five years ago. A couple of these divisions went through a conversion trauma, eventually recovered, and never undertook a subsequent DBMS project. Once was enough, they concluded.

The other divisions have continued to extend their data bases in areas of new systems development. But, this leaves a very large accumulation of computer files more or less untouched by the new technology. One manager, possibly our most enthusiastic data base advocate, believes that about 40 percent of his divisional data is now in DBMS format after some five years of development. He guesses that this figure may reach 70 to 80 percent in another five years.

We have still another group of divisions with heavy maintenance workload who have elected not to try the DBMS approach at all.

The problem is not unlike that of our Detroit skyline. We recently completed a magnificent new Renaissance Center along the riverfront, with some of the most beautiful hotel and office structures to be found anywhere. This is exciting, but there is still a long way to go to bring the rest of the city up to Renaissance Center standards. The accumulation of history is still a huge obstacle to those who want the best of things right now. Omar Khayyam, who summed up so many things in language that a sophomore can understand, expressed this frustration perfectly:

> " ... could thou and I with fate conspire
> To grasp this Sorry Scheme of things entire,
> Would we not shatter it to bits--and then
> Re-mould it nearer the Heart's desire!"

All of us, however, have to find ways to working with what we have inherited. We might all wish we could somehow get rid of the old mess and start all over again. Unfortunately, that old mess represents an investment in the hundreds of millions of dollars for my company and in many tens of billions of dollars for all of us collectively.

One of our divisions several years ago tackled this rebuilding problem in what seemed to me an innovative kind of way. This division had identified 15 overlapping files that had evolved over the years, with inventories and other data related to parts. As might be expected, there was much redundancy, and it was difficult to reconcile one file to another. A complete overhaul of the applications programs did not seem feasible, but the division hit on the idea of a DBMS master file to serve as a sort of front end to these

application programs. There was a saving of more than $100,000 annually in data preparation and data control. This limited effort brought the division into the DBMS environment and laid the basis for solid evolutionary development, including subsequent application revisions to exploit more fully the potential of the data base environment.

If there is one thing I would particularly want to see come out of this conference, then, it would be some easy-to-use, easy-to-apply, and inexpensive approaches for upgrading this great accumulation of computer files we have all been working on for the last two decades or so. We have all had tantalizing but too-brief experiences with data bases as they can be. The question before us now is, what can we do to make those benefits available wherever we need them?

Where does an organization with an accumulation of 1,500 to 3,000 programs begin, without going out of business for a year or two? What tools can it use to map out its data resources? How can it go about restructuring these files without scrapping its investment in application programs? And, even if the files can be rebuilt, what about the necessary remodeling of the interface points within the old programs? I hear that some of you came up with answers to these questions. Perhaps you can point the way to this new data-rational world we all seek.

The realization of this promise is still a long way off. The real world of tomorrow will come when the DBMS can contribute to the evolutionary process of applications development and to the full analytical use of computer resources, without exacting an extortionate price; when the DBMS can aid in the evolutionary process of hardware and software change; and when the decision to go to a DBMS is no longer a high risk affair requiring an all-out commitment through one of the most difficult conversion problems to be found in systems.

I have no doubt that something very much like this world of tomorrow will appear one of these days, in great part because groups like this one pressed on relentlessly in the definition of problems and the search for creative solutions.

3. ESTABLISHING MANAGEMENT OBJECTIVES

Richard L. Nolan

CHAIRMAN

Biographical Sketch

Richard L. Nolan is a researcher, author, and consultant in the management of information systems. As Chairman of the Nolan, Norton & Company, Inc., and a former Associate Professor at the Harvard Business School, Dr. Nolan has contributed to improving the management of data processing in complex organizations. His contributions include major publications in the areas of:

- The four stages of EDP growth
- Management accounting and control of data processing
- Managing the data resource function
- Computer data bases

Dr. Nolan's experience with the management of computer systems includes earlier associations with the Boeing Company, the Department of Defense, and numerous other large U.S. and European public corporations.

PARTICIPANTS

Marty Aronoff
Richard Canning
Larry Espe
Gordon Everest
Robert Gerritsen
Richard Godlove
Samuel Kahn
Gene Lockhart
John Lyon
Thomas Murray
Jack Newcomb
T. William Olle
Michael Samek
Steven Schindler
Richard Secrest
Edgar Sibley

3.1 OVERVIEW

> "Assimilation of computer technology into organizations is a process that has unique characteristics which management does not have a substantial base of experience to draw upon for guidance. Perhaps the most important unique characteristic is the pace of penetration of the technology in the operations and conventional information systems." [Richard L. Nolan, "Thoughts About the Fifth Stage," Data Base , Fall 1975.]

The pace of the assimilation of computer technology into the data processing organization is represented by the S-shaped "Data Processing Learning Curve."

The Data Processing Learning Curve is approximated by the growth of the data processing budget and reflects the staged evolution of the data processing environment along four growth processes:

Growth process #1. The portfolio of computer applications. The programs and procedures which are used by the organization in its business activities. The Applications Portfolio represents the cumulative end product of the data processing organization.

Growth process #2. The data processing organization and technical capabilities. The organization structures and technical capabilities found within the data processing department which are required to develop and operate application systems. These include:

- Data Processing Management Structure
- Hardware and Software Resources
- Systems Development and Operations Organizations

Growth process #3. Data processing planning and management control systems. The set of organization practices used to direct, coordinate and control those involved in the development and operation of application systems, including:

- Data Processing Planning
- Project Management
- Top Management Steering Committees

. Chargeout

. Performance Measurement

ASSIMILATION OF COMPUTER TECHNOLOGY OCCURS
IN FOUR STAGES

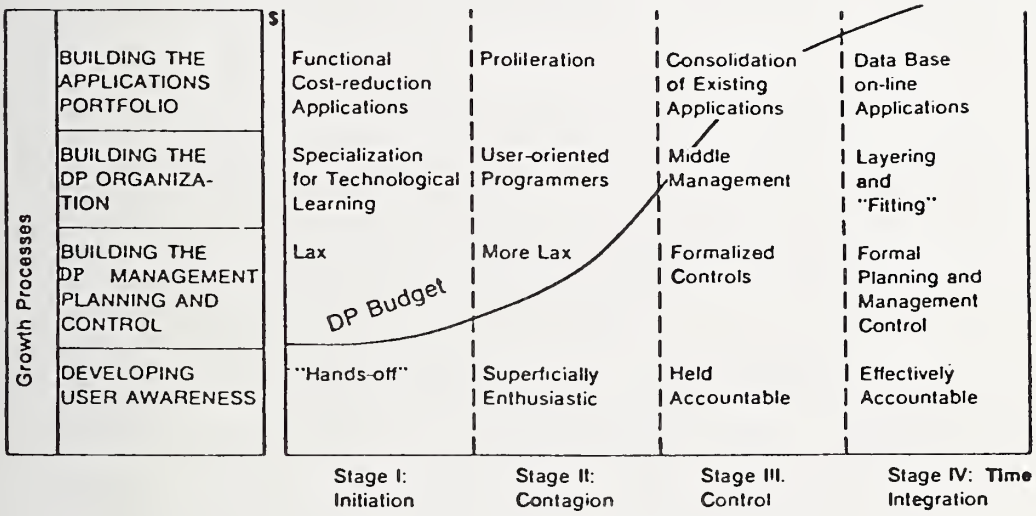


Figure 2-1
Processing Learning Curve.

Growth process #4. The user. The members of line and staff departments who must use the applications systems in order to perform their jobs.

The nature of Data Base Management Systems (DBMS) dictates that they interface with each of the four growth areas. First, the DBMS acts as the data manager for all types of application systems. Second, the DBMS introduces a

new level of technology to be assimilated by the data processing organization, and it calls for the introduction of a new data processing organization structure: the Data Base Administrator. Third, the DBMS requires that application system planning be more comprehensive and that control through "chargeout" be restructured to reflect shared resource usage. Fourth, the DBMS may impact the user by providing new functional capabilities and mechanisms for data retrieval.

Because of the integral position occupied by the DBMS in the systems environment, the conversion to data base technology and its use should be carefully managed. In managing the conversion to data base technology, data processing managers should have a well articulated set of objectives regarding each of the four data processing growth processes. These objectives should form the foundation for goals against which data base conversion activities are measured.

The initial objective of the Establishing Management Objectives panel was to analyze the impact of the data base conversion effort on each of the four data processing growth processes. Based on this analysis, the panel then determined the management considerations associated with data base conversions. The management considerations identified by the panel can be summarized into four key concepts:

- KEY CONCEPT #1: DBMS CONVERSIONS ARE A MATTER OF "WHEN?" NOT "WHETHER?"

  Conversion from a non-data base to a data base environment is a part of the natural evolution of data processing within an organization. A data processing department which has matured and progressed to a Stage III environment is typically faced with high maintenance costs and an inability to respond to ad hoc inquiries and requests from user management for integrated reports. This situation, in effect, forces the data processing department to employ data base technology to restructure the applications portfolio. In other words, conversion to a DBMS is primarily a question of how soon the data base environment should begin to be constructed, not whether a data base environment should be implemented.

- KEY CONCEPT #2: CHOOSE THE DBMS CONVERSION APPLICATION CAREFULLY

  The initial application used for conversion to data

base technology represents an important learning experience for the entire organization. As such, the initial application should:

- -be a non-trivial application
- -demonstrate the "power" of the DBMS facilities
- -be simple to avoid overextension caused by attempting to do too much, too fast

. KEY CONCEPT #3: TREAT THE INITIAL AND SUBSEQUENT DBMS CONVERSIONS SIMILAR TO OTHER SYSTEMS PROJECTS

Although a data base conversion introduces a new technology to the organization and requires the involvement of all areas in the systems environment, the risk exposure of this conversion effort can best be minimized by managing the conversion as any other large project would be managed. The same planning and justification procedures should be used. The same project management mechanisms should be exercised throughout the project life cycle. Because of the major impact caused by a data base conversion, special efforts should be made to coordinate conversion activities with steering committees, senior management, and user areas.

. KEY CONCEPT #4: PLAN AND STRUCTURE FOR FUTURE DBMS CONVERSIONS NOW; DBMS CONVERSIONS WILL BE A WAY OF LIFE

Certainly, once a DBMS is successfully installed, the conversion of applications to that DBMS will continue. However, the mature organization should also plan on converting to another DBMS at some point in time. The second DBMS may be just an enhanced version of the first DBMS, or it may be a totally new software package. In either case, it is certain that a mature data processing organization will want to take advantage of new DBMS facilities and efficiencies; therefore, DBMS conversions will become a way of life.

To prepare for these continued conversions, the data processing organization can take several steps to minimize their impact:

. minimize the application system processing logic, program code and data base design dependencies on the features of a particular DBMS

- institutionalize the Data Administration function
- fully document all business system functions on an integrated dictionary

The overall DBMS conversion philosophy developed by the Establishing Management Objectives panel can be summarized as follows:

Appreciate the technology, but recognize that DBMS conversion is a management problem.

The panel approached the topic of data base conversion in a chronological manner. As such, the following sections are organized to reflect management considerations during the life-cycle of data base conversion efforts:

- CONVERSION TO A DATA BASE ENVIRONMENT
- MINIMIZING THE IMPACT OF FUTURE CONVERSIONS
- CONVERSION FROM ONE DBMS TO ANOTHER DBMS

3.2 CONVERSION TO A DATA BASE ENVIRONMENT

A certain level of maturity is necessary before conversion to a data base environment is feasible. In general, data base technology is not appropriate for data processing departments in Stage I or early Stage II environments. With these exceptions, conversion to a DBMS should be initiated as soon as possible. However, the Stage III environment is most compatible with the initial conversion.

The following sections discuss the impact of conversion to a DBMS on the four data processing growth processes, namely:

- Applications Portfolio
- Data Processing Organization
- User Awareness
- Data Processing Planning and Management Control Systems

3.2.1 Impact On the Application Portfolio. Conversion to a data base environment will generally necessitate a substantial restructuring of the organization's application portfolio to take advantage of the enhanced capabilities of the DBMS. The several potential approaches that permit converting the application portfolio range from an evolutionary approach, in which new or replacement application systems are developed using data base technology, to a revolutionary approach, in which new development is suspended until existing application systems are converted to the new environment. Regardless of the particular approach selected, an ordering indicating a priority for conversion must be developed for the application portfolio. Within this ordering, the entry-level application is of critical importance! The above topics are discussed in greater detail in the following sections.

Approaches To Application Conversion. Two basic approaches exist for conversion of the application portfolio to a data base environment: revolutionary and evolutionary. Actually, these two approaches represent opposite ends of a continuum of approaches. No single approach is universally best. In fact, more than one approach may be operable within a given conversion; i.e., certain application systems may be converted on a revolutionary basis while other systems are converted in an evolutionary manner.

In the revolutionary approach to conversion, sometimes called resystemization, one rewrites and restructures existing application systems as necessary to operate under the new DBMS. Generally, one should avoid exclusive use of this approach for the following reasons:

- Risks overextension caused by attempting too much, too fast.
- Delays in one sub-project may impact others.
- Insufficient resources may be available for developing new systems during the conversion.

At the opposite extreme of the revolutionary approach is the evolutionary approach in which all new systems are developed under the new environment. Existing systems are not converted but rather are replaced at the end of their normal life cycle. This approach reduces the risk of overextension and the impact of delays in sub-projects. However, there are disadvantages to an evolutionary approach:

- Complex interfaces with existing, conventional systems are generally entailed.

- Local inefficiencies and redundancy typically result.

- Current organizational deficiencies and constraints may be perpetuated.

Just as the conversion of the entire applications portfolio may be approached in an evolutionary or revolutionary manner, so may the conversion of a single, existing application system. In other words, the entire application system may be converted to the new environment at one time, or the conversion may take place in phases. The latter approach, in which the reporting and update functions are converted gradually, using bridges, has the advantage of early availability of both cross-functional data and the new features of the DBMS. Furthermore, greater flexibility in scheduling the conversion is provided. However, the gradual approach has the disadvantage of redundant development and data storage, and requires increased management to provide and control the conversion bridges.

One temporary measure that may be employed to avoid or postpone conversion is to extract data from existing master files in order to build a transient, integrated data base. This data base is not maintained but instead is recreated on a cyclic basis. The data base is used for cross-functional reporting and analysis. This approach provides early availability of cross-functional data and lends itself to a specialized interrogation language. At the same time, there are certain disadvantages to this approach:

- Availability of data is achieved at the expense of redundancy and reloading.

- Problems of timeliness and consistency may be created.

- Basic application limitations are perpetuated.

Maintenance Moratoriums. There are never sufficient resources, nor is it appropriate, to permit continued maintenance and enhancements of application systems during the conversion to the data base environment. Moreover, conversion requires a relatively stationary target. Thus, a moratorium on maintenance (or more accurately on enhancement) may be declared during conversion.

The declaration of a maintenance moratorium must be the result of agreement among user, senior, and data processing management. Senior and user management support is necessary for the conversion. However, if senior management is the primary motivator behind the conversion, there will be some degree of user resistance to a maintenance moratorium. On the other hand, if the conversion is driven by user management, senior management will tolerate a moratorium on maintenance only so long as it does not interfere with normal business functions. In either case, user and data processing management must jointly determine the scope and duration of the moratorium and agree to the circumstances under which it may be modified or cancelled.

A common device for invoking moratoriums is a steering or priorities committee. Composed of data processing and user management, the steering committee is responsible for approving projects and establishing priorities. The steering committee does not manage, nor does it relieve management of its business responsibilities. Rather, it provides a forum for discussion and has power derived from its membership and sponsorship.

Analysis of Opportunities. Certain application systems indicate better opportunities for conversion than others. The following types of applications represent good opportunities for conversion:

- An application system using many different master files and/or many internal sorts, indicating the need to represent complex data structures and to support multiple paths between data.

- An application with a requirement for on-line inquiry and/or update of interrelated data. A DBMS would still be applicable, although not required, if the data were not interrelated.

- An application system with chronically heavy maintenance backlogs, suggesting redundant data and/or inflexibility with respect to its data structures.

- An application system requiring a broader view of data (either more detail or greater cross-functional breadth).

- An application which crosses functional or organizational boundaries (e.g., project control).

- An application which cannot support basic business needs.

- An application which provides data used by other systems.

Certain types of applications represent poor opportunities for conversion. For example:

- A purchased application which is maintained by a third party supplier.

- An application which uses historical data and which is processed infrequently.

- A recently installed application system which is effective in satisfying user needs.

An analysis of the characteristics of the existing applications based on the above considerations will yield a preliminary ordering for conversion of the application portfolio. As the conversion is planned in more depth, the preliminary ordering will be revised and refined to reflect such factors as precedence relationships regarding conversion, level of effort required, and the availability of resources.

Selecting the Entry-level Applications. In converting to a data base environment, a key decision is selecting the entry-level application. In an ideal world, the initial application would be selected as the vehicle for making mistakes and learning how to convert and how to manage the conversion. It would have a low profile and not present any risk to the business. However, the realities of the world will force initial conversion of a system which has visibility, contains some element of risk, and which must be completed quickly. The factors listed below should be considered in identifying the best opportunity for developing technical competence while simultaneously reducing risk and visibility:

- The application should be representative and non-trivial.

- It should be a good DBMS application (though not necessarily the best).

- It represents a relatively low risk to the business.

- It provides sufficient opportunity for learning.
- It is either an old system or technically obsolete.
- It provides eventual visibility as a vehicle for management controls.
- It is "owned" by a vocal, important, but neglected (by data processing), segment of the business.

3.2.2 Impact On the EDP Organization. This section discusses the following topics relating to the impact of conversion on the data processing organization:

- Organizational considerations.
- Technical aspects: tools and methodologies.
- Data processing personnel skill requirements.

Organizational Considerations. Converting to a data base environment generally entails reorganization of the data processing function in order to provide the technical and administrative means for managing data as a resource. A key organizational consideration is the need to establish a Data Base Administration (DBA) function within data processing. Conversion will also impact the applications development and computer operations functions within data processing.

Data base administration. The Data Base Administration (DBA) function is responsible for defining, controlling, and administering the data resources of an organization The many responsibilities of the DBA function are not discussed in detail here since they are covered extensively in the literature. However, some of the major responsibilities include the following:

- Data base definition/redefinition. DBA must have primary responsibility for defining the logical and physical structure of the data base, not merely consulting responsibility.
- Data base integrity. DBA is responsible for protecting the physical existence of the data base and for preventing unauthorized or accidental access to the data base.
- Performance monitoring. DBA must monitor usage of the data base and collect statistics to determine the efficiency and effectiveness of the data base in satisfying the needs of the user community.

- Conflict mediation. DBA must mediate the conflicting needs and preference of diverse user groups that arise because of data sharing.

Many alternatives exist for locating the DBA function within the overall corporate structure. Three such alternatives include the following:

- Within the data processing organization. In order to avoid an application orientation or an emphasis on computer efficiency, DBA should, in general, not report to Applications Development or Computer Operations, respectively. Rather, DBA should report to the highest full-time data processing executive.

- Corporate level. When located at the highest corporate level, DBA can take a broad view of data as a corporate resource. Furthermore, DBA is in a position to resolve conflicts between user areas. When DBA resides at this location, some of the more technical aspects of the DBA function are typically performed within the data processing organization.

- Matrix organization. This structure is patterned after the aerospace industry where a given project draws upon all functional areas. In this case, the DBA staff would report functionally to DBA but would also report directly to a project manager. This organizational strategy has the advantages of recognizing the integration required for a data base, puts DBA at an equal level with other functional areas, and serves to increase communication during application development.

Within the DBA function, the two basic organizational strategies are functional specialization versus application area specialization.

- Functional Specialization. This strategy organizes DBA according to functions performed, such as data base design, performance monitoring, data dictionary, and so on. This approach has the disadvantage of ensuring that no one person is knowledgeable about all aspects of DBA support for a particular application system.

- Application Area Specialization. In this approach, one person within DBA is responsible for performing all DBA functions for a particular application area, including both application development and operation. This approach has the disadvantage of developing expertise within functional areas of DBA

more slowly. Furthermore, unless controlled, activities within DBA may become fragmented. However, this approach results in an interesting and challenging job and facilitates attracting and keeping capable personnel.

Applications development. Conversion to a data base environment will affect the applications development function within the data processing organization in several ways. The most fundamental impact upon applications development will be the change from an applications orientation to a data orientation. Conversion to a data base environment should also broaden the scope of the application developers. Specifically, the developers need to understand the basic business processes and to develop application systems that cross organizational boundaries.

The application development methodology will have to be modified by delimiting the relative responsibilities of both DBA and applications development. Moreover, the basic approach to application development may be revolutionized as a result of conversion to a DBMS. Specifically, instead of a rigorous approach to application development, the DBMS may permit an iterative or convergence approach. With this approach, user requirements are not defined in detail before developing the application system. Rather, user requirements are defined at a more general level and a system is quickly built using the DBMS. When presented with the system outputs, the user specifies any required changes, which are then incorporated into the system. This process is repeated until the application system satisfies user needs. Note that this approach to application development requires a DBMS in which data base definition, creation, and redefinition and report writing are quickly and easily accomplished.

Computer operations. Conversion to a data base environment will impact the Computer Operations function in two ways. First, many of the responsibilities of computer operations function will be transferred to the newly established DBA function. Second, the characteristics of the application systems may change. Specifically, the DBMS may facilitate the development and operation of on-line applications as opposed to the more traditional batch systems. Consequently, the computer operations function may have to reorganize to operate within this more dynamic environment.

Technical Aspects -- Tools and Methodologies. Because the subject of DBMS selection has been covered adequately in the literature, it was not addressed by this panel. However, the following are some of the tools and methodologies typically required in making effective use of

the DBMS after its installation:

- Data dictionary/directory. A tool for organizing, documenting, inventorying, and controlling data. It provides for a more comprehensive definition of data than is possible in the DDL facility of most commercial DBMS's. As such, it is essential for management of data as a resource.

- Data base design and validation tools. Used to facilitate the design process and to validate the resultant design prior to programming. Included in this category are such tools as hashing algorithm analyzers and data base simulation techniques.

- Performance monitoring tools. Useful in analyzing and tuning the physical data base structure. These tools provide statistics on data base usage and operation.

- Application development tools. Used to facilitate the development of application systems, including such tools as terminal simulators which operate in batch mode and test data base generators.

- Data base storage structure validation utilities. Used to verify that a stored data base conforms to its definition or to assess the extent of damage of a damaged data base. Examples include a "chain walker" utility.

- Query/report writer facility. Enables users to access the data base and extract data without having to write a procedural program in a conventional programming language.

- Data base design methodology. Needed to standardize the approach to data base design and to provide guidance in using the data base design, modeling, and monitoring tools.

- Application development methodology. Specifies the standardized approach to developing application systems; i.e., the activities to be performed during the development process and the corresponding roles and responsibilities of each of the various project participants. Of particular importance is the need to define the points in the development process at which DBA and applications development functions must interface and the relative responsibilities of each with respect to application development.

- Documentation methodologies. Needed by DBA to document data definitions uniformly and to document data base design decisions.

Data Processing Personnel Skill Requirements. The impact of conversion to a data base environment on skill requirements will be considered in this section.

Types of skills. The following types of skills are needed in a data base environment:

- Data Base Administration. DBA should be staffed with individuals who are strong technically, interface well with people, and collectively are knowledgeable about the DBMS itself, the tools necessary to support it, the application development process, and the corporation and its data.

- Logical data base design. Within DBA there is a need for individuals possessing the ability to recognize and catalog data elements, to group related data elements, identify relationships between groups, and to use the data description language.

- Physical data base design. Within DBA there is a need for individuals knowledgeable with respect to organization techniques, data compression, trade-offs in data base design, simulation, and modeling techniques.

- DML programming. DBA should include individuals with knowledge of the DML and its associated host language, data base navigation, and the currency concept.

Acquisition and training. Obviously, the required skills may be developed internally or acquired externally. Hiring the required personnel has the advantage of bringing experience and new ideas into the data processing organization. However, individuals knowledgeable with respect to DBMS are scarce and hence expensive. Moreover, individuals brought in from the outside typically have little, if any, knowledge of the business.

Developing skills internally has the advantage of building DBMS skills on top of knowledge of the business. Furthermore, control can be exercised over what is learned and when. Finally, it is generally less expensive and disruptive than hiring.

When skills are developed internally, there are several possible approaches to training:

- In-house. In this approach, staff personnel possessing the necessary skills teach these skills to others by means of courses or joint projects. This approach may fit well with initial application development and has no cash cost. Moreover, the mere act of having to teach their skills to others enhances the knowledge and understanding of the teachers themselves. There are several disadvantages to this approach: it requires the time of the most capable personnel when they may be more effectively used elsewhere; it cannot be used where the required skills do not exist internally; as a closed system, it excludes differing points of view.

- Vendor. This approach utilizes the courses offered by DBMS and support software vendors. Vendor courses may be a relatively inexpensive approach, particularly when courses are bundled as part of the purchase/lease price. Furthermore, the internal staff are likely to benefit from the expertise of the vendor. However, the courses may be only a thinly-disguised sales pitch.

- Other approaches. Additional approaches to training include:

  -independent educational organizations

  -colleges or universities

  -videotape/cassette courses

Turnover. Conversion to a data base environment may result in employee turnover. The new DBMS may be perceived by the staff as being threatening and, hence, may be resisted. This resistance to change may be overcome somewhat by involving the staff in the series of decisions leading to the acquisition of a DBMS. If required skills are obtained through hiring, the existing employees are likely to resent the high salaries paid to the new employees. Finally, as the skills of the staff increase, so does their market value and it becomes increasingly expensive to retain the staff. These three factors -- resistance to change, resentment of new hires, and increased employee market value -- tend to increase turnover following conversion to a DBMS environment.

On the other hand, certain factors tend to decrease turnover. Specifically, conversion to a data base environment involves new opportunities for individual growth and excitement such as new technology, new hardware and software, and major development efforts. Properly exploited, these factors can increase job satisfaction and correspondingly decrease turnover.

3.2.3 Impact On Planning and Control Systems. With the exception of the chargeout mechanism, conversion to a data base environment will not affect the basic mechanisms for planning and control. However, recognize that conversion is itself a process to be managed. This entails applying justification procedures for conversion, planning the conversion, establishing review and approval checkpoints, and monitoring progress.

Planning the Conversion. Planning for the conversion requires the involvement of senior, user, and data processing management:

- Attempts to convert to a data base environment without senior management support runs a high risk of failure. If senior management has not formally authorized DBMS studies or incorporated DBMS planning into corporate plans, the probability of successful conversion is remote.

- Conversion will have a significant impact on user departments in the form of disruption of normal data processing services, restructuring of application systems, and a change in orientation on the part of users from ownership to sharing of data. Consequently, user involvement in planning the conversion is critical.

- Conversion to a DBMS generally affects the structure, system development methodology, personnel skill requirements, and hardware/software configuration of the data processing function. The lead time necessary to develop the appropriate infrastructure for operating in a data base environment must be appreciated and planned for accordingly.

Given senior management support for the conversion, one strategy for obtaining the required involvement in the planning process is to establish a steering committee for the data base as mentioned in the earlier section on Maintenance Moratorium. This steering committee contains representatives from both user departments and from data processing and is responsible for controlling the evolution

of the data base. As such, the Data base Steering Committee is subordinate to the data processing Strategic Steering Committee, which is concerned with the evolution of the entire data processing function within the enterprise.

Given the appropriate participation, a necessary first step in converting from a non-data base to a data base environment is the development of an architectural plan for the data base. This plan describes the intended structure of the target data base. Conceptually, a data base represents a model or image of the organization which it serves. In order for the data base to represent an accurate image of the organization, it is necessary for the data base structure to reflect the fundamental business processes performed in the organization. Consequently, the designers of the data base must first understand the key decisions and activities required to manage and administer the resources and operations of the enterprise. This typically entails a cross-functional study of the enterprise in order to identify the business processes and information needs of the various user departments.

The architectural plan permits planning and scheduling the migration of application programs, manual procedures, and people to a data base environment. This implementation plan must incorporate review and approval checkpoints that enable management to control and monitor the conversion process.

Controlling the Conversion. The actual conversion to a data base environment is effected by a project team composed of representatives from user departments, applications development, and data base administration. At formally established checkpoints during the conversion, the data base steering committee reviews the progress of the project. Items reviewed and analyzed include the following:

- Projected benefits vs. actual benefits
- Data quality (i.e., completeness, timeliness, and availability)
- Projected operating and development costs vs. actual costs
- Actual costs of collecting, maintaining, and storing data vs. benefits realized

- Project performance (i.e., performance of the project team against the conversion schedule and budget)

Based on the review, the data base steering committee takes the appropriate approval action (e.g., go/no go) with respect to the conversion activity.

Chargeback Considerations. The costs of operating in a data base (i.e., shared data) environment are extremely difficult to charge back to individual users in an equitable manner. At best, complex job accounting systems can only approximate actual resource usage. Moreover, the chargeback algorithm must not be dysfunctional with respect to its impact on the various user departments. Conversion to a data base environment frequently requires that a user department supply data which it does not itself use. The chargeback algorithm must reward, not penalize, such behavior on the part of the user department.

Some considerations in developing an appropriate chargeback algorithm include the following:

- Consider capitalization of the costs of conversion instead of treating such costs as current expense in order to avoid inhibiting user departments from undergoing the conversion.

- User departments typically have little control over the costs of conversion. Consequently, consider treating such costs as unallocated overhead, since allocation will have little effect on the decisions or efficiency of the user departments.

- Because ongoing costs of collecting, maintaining, and storing data are difficult to associate with individual users, consider developing percentage allocation factors for these costs based on periodic reviews of data base usage. Alternatively, consider treating these costs as overhead.

- Resource usage for retrieval and processing purposes are easier to approximate, and such costs should be charged directly to the users.

- Consider incorporating a reverse charging mechanism into the chargeback algorithm in order to compensate users who supply data which they do not use.

3.2.4 Impact of Conversion On User Awareness. The most fundamental impact of conversion to a data base environment is the required change in orientation on the part of users. No longer are files and applications "owned" by a particular user department. Rather, data must be viewed as a corporate resource to be shared by all user departments. The requirement for sharing constrains the freedom of a user to change arbitrarily and unilaterally the definition of the data.

Sharing of data will impact users in a second way. Following conversion to a data base environment, users may be required to supply data which they themselves do not use. As already discussed, the chargeback algorithm must be structured to reward such behavior. Furthermore, suppliers of data in general must be infused with a sense of responsibility (not ownership) for the data in order to maintain data quality.

Conversion to a data base environment may impact the user community in other ways:

- Planning the conversion. User participation in developing both the architectural plan and implementation plan is necessary in order to obtain user commitment and to ensure that the resultant data base satisfies user needs.

- Disruption of operations. Normal data processing services are likely to be severely disrupted as a result of such factors as limited availability of personnel and maintenance moratoriums. Furthermore, the conversion may disrupt and strain user operations as new and old applications are operated in parallel.

- Resolution of inconsistencies. Creation of the data base typically entails merging of application-oriented files. During this process, inconsistencies in both data definitions and data values are identified. These inconsistencies must then be resolved by the relevant user departments.

- Structure of user department. The structure of a user department may no longer be effective following conversion; e.g., the user department may be designed around a particular application system. Restructuring application systems during conversion may precipitate user reorganization.

- New organizational roles. Conversion may cause new organizational roles to evolve in user departments. For example, in order to provide coordination between Data Base Administration and the user departments, a "user data administrator" may evolve. The user data administrator serves as the focal point for participation and involvement on the part of the user department both during and subsequent to the conversion.

- Systems analysis. By providing such tools as a high-level query language and/or a generalized report writer, the DBMS enables non-technical users to access the data base directly; i.e., users are less dependent upon programmers to satisfy simple requests for information. This increased availability of data may result in the migration of the systems analysis function from data processing to user departments.

- Personnel skill requirements. Conversion may impact the skill requirements of user personnel. For example, conversion to a data base environment may also result in operating certain application systems online, requiring that the user department acquire or develop terminal operations skills.

3.3 MINIMIZING THE IMPACT OF FUTURE CONVERSIONS

The initial conversion to a data base environment is not likely to be the only data base-related conversion that an enterprise will undergo. Rather, conversions of one form or another are likely to be a way of life. However, there are certain measures that the data processing organization can take to minimize the impact of future conversions, including:

- Institutionalization of the Data Base Administration function.

- Insulation of programs from a particular DBMS.

- DBMS independent data base design.

3.3.1 Institutionalization of the DBA Function. A well-established DBA function will minimize the impact of future data base conversions. Specifically, the DBA function should take action as follows:

- Maintain data definitions and relationships in an up-to-date data dictionary.

- Document the structure and contents of all data bases independently of the data description language of the DBMS.

- Develop methodologies and standards for data base design which are independent of any particular DBMS.

3.3.2 DBMS Independent Data Base Design. The recent work of the ANSI/X3/SPARC Study Group on DBMS introduced the idea of a conceptual data model. The topic has also been pursued extensively in research papers. In practical terms it amounts to design and documentation of the data base in a form independent of any particular DBMS. In translating the conceptual model to the data description language of the DBMS selected for implementation, the design decisions which are predicated on the characteristics of the DBMS are more clearly distinguishable from the natural structure of the data. Should the DBMS be changed subsequent to implementation, it is possible to focus more clearly on the structural conversions required of the data base and the applications accessing the data base. As a point of interest, use of the conceptual data model is appropriate whether a DBMS or conventional files are to be used.

3.3.3 Insulate Programs From the DBMS. Two sets of circumstances may motivate an organization to attempt to insulate its application programs from any one DBMS. On the one hand, an organization may be unwilling to commit completely to the use of a particular DBMS. Rather, it may desire to keep its options open with respect to converting to a different DBMS at a later date. Alternatively, a large multi-division corporation may desire to develop common application systems for the divisions; yet it may find that the data processing organizations within autonomous divisions have installed different DBMS's.

It is possible to build an interface between application programs and the DBMS in order to isolate the programs from the DBMS. That is, the programs do not interact directly with the DBMS. Rather, standard program requests for DBMS services are translated into the required DML statements either at compilation time or at execution time. Thus, application programs are insulated from changes in the DBMS as long as an interface module can be developed to translate program requests into the DML statements of the new DBMS. Similarly, a single application will execute under any number of DBMS's as long as the appropriate interface modules exist. Furthermore, the multi-division corporation retains the flexibility of standardizing on a

single DBMS at a future date. The negative aspects of this approach include reduced system efficiency and the possibility of ending up with a pseudo-DBMS whose capabilities represent the lowest common denominator of the various DBMS's for which interfaces are built or planned. Nevertheless, a number of corporations worldwide have adopted or are adopting this approach.

3.4 CONVERSION FROM ONE DBMS TO ANOTHER DBMS

As a data processing organization goes through the experiential learning necessary to assimilate data base technology, the functions and features of the data base management system package currently installed will tend to be more highly utilized. Users will have positive experiences with the facilities offered by the DBMS and will subsequently place greater burdens on those facilities. Also, the technical capabilities of the DBMS will be increasingly utilized by the data processing staff in order to meet user requirements.

In short, the tendency to use the full functions of the DBMS over time will place a strain on the capabilities of the DBMS. This is manifested by either decreasing systems processing efficiency or increasing effort necessary to develop systems which meet user needs. These increased costs are recognized by both users and data processing personnel who then initiate a search for increased DBMS capabilities and, thus, begin data base conversion effort.

This second type of data base conversion can be characterized by either a complete change in data base management system packages or an upgrade in the version of the DBMS currently installed. This section discusses the impact of the conversion from one data base system environment to another on each of the four growth processes previously discussed. The section is organized as follows:

- Reasons to go through the conversion.
- Economic considerations of the conversion effort.
- Conversion activities and their impacts.
- Developing a strategy for the conversion.

3.4.1 Reasons for Conversion. As has been previously discussed, the most prevalent reason to undertake a conversion from one DBMS to another DBMS-1 to DBMS-2 Conversion is to install a better DBMS. A better DBMS is usually defined as having:

- Improved functions (more complete).
- Better performance.
- Improved query capability.
- Development of richer data structures.
- More efficient usage of the computer resource through decreased cycles and/or space.
- Improved or added communication functions.
- Availability of transaction processing.
- Distributed processing capability.

Another major reason to undertake a DBMS-1 to DBMS-2 Conversion is to standardize DBMS usage within the company. Many large corporations are finding that the DBMS selections made several years ago to meet specific application needs have resulted in the installation of several DBMS packages within the data processing organization. The impact of multi-DBMS usage in a single data processing environment is major. For example:

- Application programs are constrained to the design and processing characteristics unique to each DBMS.
- Data files are structured to be accessed by a single DBMS.
- Design and programming personnel develop the skills necessary to implement systems associated with a single DBMS.

The multi-DBMS environment results in a substantial investment in data processing personnel technical skills and reduces the potential for integrating applications that operate on different DBMS's.

For these reasons many companies are now developing standards for data base management system usage. Those standards are usually application systems to be developed under a single DBMS. Exceptions may exist where the application to be developed is stand-alone in nature with a

low potential for integration with other systems.

The last major reason for DBMS-1 to DBMS-2 conversion is that such a conversion is dictated by a hardware change. Many of the commercially available DBMS's are offered by large mainframe vendors. As such, a move from one hardware vendor to another will necessitate a change in DBMS usage. This can become quite a complex effort in that the source code and data base storage structures of all programs will require changes. If there is a history of hardware conversions in the company, the wise data processing manager should select a DBMS that is not hardware dependent.

3.4.2 Economic Considerations. A key concept that was introduced in the previous section is that the data base conversion effort should be analyzed and managed like any other high-risk systems project. The same concept applies to a DBMS-1 to DBMS-2 conversion effort. As a result, the DBMS-1 to DBMS-2 conversion should be justified on the same basis as any other systems development effort is justified in the company. An economic justification should be made on the basis of costs and benefits associated with the data base conversion. The economic justification is particularly important if the major reason for the data base conversion is either better DBMS or standardization of DBMS usage. For a hardware change, the cost and benefits associated with the DBMS conversion should be included in the justification for the hardware change.

The economic justification for a DBMS-1 to DBMS-2 conversion should be based on a succinct articulation of the COSTS and BENEFITS directly associated with the conversion effort. Costs should be identified on an incremental basis and be classified into three categories:

- One-time conversion costs.
- Incremental costs for each planned application to be converted to the
- new DBMS.
- On-going DBMS support costs.

Benefits should likewise be identified on an incremental basis within the same time frames as the associated costs. Benefits are divided into two categories:

- Discernible/definable cost savings in development, maintenance, and operations.

. Intangible cost savings.

A more complete description of the types of economic considerations to be addressed is contained in DATA BASE DIRECTIONS: THE NEXT STEPS, National Bureau of Standards Special Publication 451.

Above all, the justification for a data base conversion should be developed and communicated to management in the same manner that any other project is justified.

3.4.3 Conversion Activities and Their Impact. The impact of a DBMS-1 to DBMS-2 conversion effort can be felt on each of the four growth processes previously discussed. Many of the types of impacts are the same as those previously identified in a conversion to data base technology. Users and data processing management should recognize that many of the same experiential learning processes occur in subsequent data base conversions as they do in the initial effort.

Impact On Application Portfolio. The impact of subsequent data base conversions on application portfolios occurs in three areas:

. Application programs.

. Data bases.

. Catalogued modules.

Application programs. Because application programs are buffered from actual data storage structures by the DBMS, the unique characteristics of each DBMS will have a major impact on application programs in the following areas:

. DBMS "call" structures.

. Programs view of data and mappings (model, structure, content).

. Application program logic.

. Data communications.

Data bases. Physical data storage structures and logical data relationships are implemented via unique DBMS utilities and are patterned after distinct DBMS requirements. As such, data bases developed under one DBMS are not readily accessible by other DBMS packages. Specifically, data bases are impacted by the vagaries of data base management systems in the following ways:

- Data base definitions in both the DBMS and in the Data Dictionary.
- Data content and storage format.
- Use of data base design and simulation aids.
- Conversion aids.

Catalogued modules. Though processed just as any other program, catalogued modules differ from application programs in their function and method of development. The specific types of catalogued modules which are impacted by a change in DBMS are:

- Catalogued queries.
- Catalogued report definitions.
- Catalogued transaction definitions.

Impact On the Data Processing Organization. As was previously discussed in the section on conversion to the data base environment, the major impact on the data processing organization structure is the implementation of the Data Base Administration organization. Because this DBA structure has already been integrated into the data processing environment during the initial data base experience, the conversion from one DBMS to another will not have a major impact on it. Only procedural fine-tuning will be required as the functions of the DBMS change. However, it should be recognized that a substantial learning curve will likely exist as the new DBMS technology is assimilated by DBA personnel. Other organizational structure changes in the data processing environment as a result of the DBMS-1 to DBMS-2 conversion will be minimal.

The major organizational impact throughout both data processing and users areas is likely to be in the technical and functional education required before, during, and after the conversion effort. Data processing and user personnel in all areas of systems development and operation will have to be trained on the new aspects of the DBMS. Training programs for all people should be identified and initiated in advance of the conversion implementation.

Another major area of impact on the data processing organization from a data base conversion is the modifications in documentation necessary to accommodate the new DBMS environment. Changes in documentation will occur in the following areas:

- DBMS functional and technical support (reference) documentation.

- Functional and technical descriptions of any application systems converted onto the new DBMS.

- Physical and logical descriptions of any data bases converted.

- User-oriented descriptions of application systems processing characteristics.

- System development methodology documentation that references particular aspects of data base or application development.

Impact On Data Processing Management Control Systems. As previously discussed, Data Processing Management Control Systems comprise those sets of procedures regularly used to control both systems development and operations functions. The conversion from one DBMS to another is not going to modify the conceptual framework used to control the data processing environment. However, specific changes will affect the mechanics of control:

Data processing budgeting and user chargeback. The chargeback algorithm used to charge users for data processing services is likely to change because of modifications in:

- DBMS overhead (cycles).

- DBMS space requirements.

- methods of implementing logical relationships.

- ownership of data items.

- differences in efforts required to design and implement application systems.

- differences in methods used to structure ad-hoc queries and periodic reports.

- methods of charging end-user cost centers for the one-time costs of conversion. The time-frame of allocating these charges can also be important (one lump sum vs. periodic payments).

Systems development methodology.

- There will be changes in the time frame and types of effort required in systems development.

- Conceptual approach to developing systems may change due to total effort or time frame required to generate sample reports on test data bases.

- Design procedures in the methodology not likely to change if DBMS facilities are similar; only jargon should change in documentation.

Data processing performance measurement.

- Systems development and operations standards by which data processing personnel are regularly measured should change due to new functions and especially to a new learning curve.

- Computer operations performance monitors and standards will change due to new processing technology.

Integrity control.

- Adequate data base backup should be carefully analyzed and managed during the conversion process.

- Operational restart/recovery procedures will change due to new DBMS functions or utilities.

- Processing of data exceptions may differ.

Security control.

- Differences in methods of data access security should be recognized.

- Data manipulation restrictions may vary from DBMS-1 to DBMS-2.

Privacy control.

- Where appropriate, special care should be taken that all privacy disclosures are logged during a data base conversion per recent government regulations.

Impact On User Areas. A key note of data base conversions is that the conversion should be as transparent as possible to user areas. This axiom holds that the processing impact on user areas should be held to a minimum and that the necessary technical capabilities to support the conversion should reside in the data processing area.

The impact of the data base conversion effort should be readily apparent to users regarding:

- Functional improvements (e.g., new query language).
- Increased data content (e.g., "while we are changing, let's add ...").
- Increase in sharing of data will highlight data inconsistencies, validations, and format errors.
- Possible planned disruption of services during conversion period.
- Data ownership changes.
- User mental images or expectations may change.
- Archival data capabilities may change (e.g., meeting the needs of IRS, EEO, etc.).

3.4.4 Developing a Conversion Strategy. The well-managed data processing installation should carefully articulate a data base conversion strategy and plan before initiating any conversion effort. Specifically, the data processing management personnel should do the following:

- Determine specific conversion objectives.
- Analyze pros and cons and develop a memo of rationale.
- Develop a conversion strategy.
- Develop a detailed plan for procedures, data, and programs.

The following is a list of possible strategies which may be undertaken:

- Unbundle conversion of procedures, programs, and data (conversion may affect all)
- Build bridge from old programs to new data structures
- Convert non-vital program first
- Migrate data and program conversion as maintenance threshold is approached

- Consider bringing in outsiders to convert procedures
- Run parallel for vital systems:
  - -frustrated users may be a problem.
  - -you will probably be turning up old bugs.
  - -parallel may be security blanket only.
- Avoid parallel running where possible by careful planning and phased cutover or "fast" cutover with fall back contingency plan, but note the possibility of high risk exposure.
- Map how all users use the shared resources.
- Split input stream between old and new applications and plan valildity checks carefully.
- Use DBMS facilities such as dumping/loading and mapping, if available.
- Convert data all at once, then convert programs as needed.
- Restructure data if necessary before any program conversion.
- Benchmark first.

Once the appropriate conversion strategy is determined, a detailed conversion plan should be developed. The following is a list of considerations to reference when developing a conversion plan:

- Develop a work plan as in any other systems development effort.
- Append original feasibility documents and review in the light of latest detailed specifications for conversion.
- Re-evaluate "Go/No Go" at pre-defined checkpoints in conversion process.
- Subject the conversion process to standard project management control.

- Develop a contingency plan.
- Use planning document as an education tool for project personnel.
- Evaluate and select conversion aids.
- Flesh out impacts (see section on Conversion Activities and their Impact above) into manageable tasks. Schedule these tasks and establish a reasonable work breakdown structure.
- Go through planning document with an implementation hat on.
- Control project on at least a weekly basis.
- Plan for heavy user involvement in data translation to resolve inconsistencies and consolidate validation checks.
- Involve external and internal audit staffs.

3.5 SUMMARY

In summary, the panel on Establishing Management Objectives analyzed the impact of a DBMS conversion in terms of the four growth processes along which the D.P. function evolves, namely:

- the portfolio of computer applications
- the D.P. organization and its technical capabilities
- the D.P. planning and management control systems
- the user

Two types of DBMS conversions were addressed: (1) the initial conversion to a data base environment, and (2) the conversion from one DBMS to a second DBMS.

Four key concepts summarize the findings of the panel:

- Conversion to a data base environment is primarily a question of how soon the data base environment should begin to be constructed, not whether a data base environment should be implemented

- Because the initial data base application represents an important learning experience for the entire organization, great care must be exercised in selecting the entry-level data base application

- The exposure to risk associated with a DBMS conversion is minimized by managing the conversion as any other large systems project

- An organization will likely be involved in multiple DBMS-related conversions, from the initial conversion of conventional applications to data base technology to the conversion of applications from one DBMS to a second DBMS. Accordingly, the D.P. organization should plan and structure for future DBMS conversions now in order to minimize the impact of future conversions

In closing, appreciate the technology involved, but recognize that a data base conversion is a management problem.

James H. Burrows

CHAIRMAN

Biographical Sketch

James H. Burrows is Director of the Institute for Computer Sciences and Technology within the National Bureau of Standards. At the time of the workshop he served as Associate Director of Data Automation, USAF and was responsible through the Director for all data automation matters within the Air Force. He directed, controled, and managed the Air Force's overall data automation program. Prior to that position, he was Technical Director for Command and Information Systems Division and Department Head for the Computer Applications Department at MITRE.

He is a graduate of the United States Military Academy and has degrees from MIT and the University of Chicago. He is a member of ACM, Institute of Management Sciences, AAAS, and American Society of Public Administration.

Participants

Edward Arvel
Michael Carter
Joseph Collica
Elizabeth Courte
Ahron Davidi
Ruth Dyke
Halaine Maccabee
Steven Merritt
Alfred Sorkowitz

4.1 INTRODUCTION

The members of this panel reviewed their experiences under various conversion scenarios to identify the critical factors which led to success or led to delays and failure. The panel felt that the resultant list would benefit managers facing data base conversion projects. While it would have been noteworthy if some set of tools, methodologies, etc. had emerged from the experience of the group which would make any future conversions easy and riskless, the panel found none. In the panel's opinion, a successful conversion takes tedious preplanning and careful execution; and in the current state of practice no known panacea exists.

This panel will not try to tell you how to justify data base conversion but will give its best advice on what to consider when approaching the conversion task. The panel began its considerations with the assumption that the justification had been previously determined. The panel agreed that its experience proved conversion justified. In fact, for some panel members conversion was unavoidable. Others felt they had demonstrated the value of conversion to DBMS but with some reservations. In either case, our experience may help you.

TEN CONVERSION EXPERIENCES		
Scenarios:		
Manual to DBMS	2	
File to DBMS	6	1 involved machine change
DBMS1 to DBMS2	2	1 involved machine change
Hardware:		
IBM -Univac - Honeywell		
DBMS Packages:		
IMS - TOTAL - DMS 1000 - S2K - IDS I & II		

Table 1: Conversion Experiences

The panel consisted of practitioners, people who fought in the trenches and who made or followed decisions to use DBMS techniques in actual situations--and who, in most cases, had to suffer the barrage of consequences.

The group reviewed only recent experiences, although several participants have been using the concepts and techniques since the early 60's. Table 1 presents a summary of the types of conversion scenarios reviewed, the machines involved, and the DBMS variety.

Specific details and guidelines that resulted from the review of these experiences during the workshop are detailed below in the section called, "ANNEX: CONVERSION EXPERIENCES."

4.2 PERSPECTIVES

During the course of its discussion, it became clear that the panel and the people with whom the panel dealt saw DBMS technology, its promises, and its threats differently. One of the simplest views sees a batch system with an on-line access method to allow queries and, possibly, on-line data-entry, e.g., a permanent operation using a leased national timesharing network for access during the day with overnight update on owned facilities. One might use this operational mode as an educational step in the process of selling and orienting the user and technical personnel to the power and first level details of DBMS or as an intermediate step in transitioning from one DBMS to another. Another view saw the DBMS as a "super access" method, promoting fast response and reducing the manpower costs of responding to new requirements. A third view saw data base systems as the best way to permit ad hoc on-line access and a fourth saw them providing multi-file processing with minimal expenditure of technical talent.

However, all of the above were considered subsidiary to a view that DBMS controls the growth of information in support of the corporate business activities--with less pain and suffering for all participants.

Many also see DBMS as a new sequence of buzz words and acronyms: schema, root segment, subschema, DDL, DML; a new mystique; a new barrier between the ADP community and the real world. As an industry, we may yet learn that grist for the Ph.D. mill and true payoff to the business world are not necessarily the same. But, the boss knows it and suspects these new "academically sponsored" techniques. Who else invents language sounding like that above?

And as usual, the new technology threatens the current technologists.

Many of those selling DBMS emphasize the "total" commitment to the integration possible using a DBMS. This convinces management that DBMS, more than a new risk, means "total" risk. Who would make a decision to so "expose" the company? Many managers hear about the attributes of DBMS that make it seem a new technology that looks good for everything. But managers have seen other new technologies with similar histories. They share with DBMS the following traits:

- More variants seem to appear daily
- No one seems to be in charge of standards, or even have a clue on how to start
- What is available on the market is of uneven quality--ease of use, reliability, power and vendor support seem to be random variables
- No accepted guidelines on "how to use" and "what to avoid" in the new technology seem to exist. This disconcerts managers.
- The technology conflicts with other forces. In DBMS the basic principle is this: the corporation should consider its data as a central, corporate resource, not a private resource of each local subunit. At the same time, it is clear to the large national and international corporations that local authority is the only way to inspire local responsibility.
- Confusion over the fundamental purpose of the new technology. It is not clear whether the advocates of DBMS are technologists trying their best to meet requested/perceived needs or management specialists who see an opportunity to enforce central control. In any case, DBMS technology claims advantages for the user whether the corporation pursues centralized, decentralized, or distributed responsibility, authority, or data processing.

4.3 FINDINGS

The panel found management's attitude about data processing shifting to concentrate on the data aspects. Data is not a private resource for local exploitation. Therefore, management must commit to both plan for and control the use of the data.

The first commitment, to plan for the use of data across the corporate body, is clearly a new job that is beyond the purview of any single department and also beyond the authority usually vested in the data processing department. Anyone trying to do central planning faces a struggle because each private domain will resist change.

Not only a new agent, but also a new function is needed for central planning: Data Base Administration. Performed at two levels, this function may not be under a common manager. The first level is development/design oriented. This group designs the data base, makes available the appropriate tools and utilities, analyses the usage of the data base(s), and restructures the data base. This group also establishes the procedures for the second level, the operation-oriented Data Base Administrators.

These operation-oriented Data Base Administrators must deal with the physical aspects of the data base. Such old concepts as allocation of storage, protection, recovery, dumps, verification, etc., are performed in quite different ways under a DBMS.

Typical figures for the size of the DBA group may fall between 4 and 30. Only in very small installations, is it a one man job.

The panel unanimously warned against an overambitious project size for the initial data base application. This usually results in errors in cost and time estimates. The anxious user withdraws to a "wait and see" mode and announces changes in his needs as the project progresses. The too-large project will probably be unable to cope with such changes, even if the original goals are successfully delivered. All of this leads to alienation of the users and managers and a distinct hesitation to continue with the new application technology.

The panel noted the availability of packages at many levels. However, knowledge of how to best choose and use a data base system is in short supply in any organization about to enter the data base world. Help from both consultants and suppliers is available and should be used.

4.3.1 Industrial/Governmental Practices.

The deliberations on pre-planning and planning identified a dichotomy between private industry and the Federal Government in both the processes controlling and the results accomplished.

Industrial/Commercial (I/C) practice, when making decisions on hardware and support software (DBMS, Telecommunications, etc. packages), requires a study of the alternatives followed by negotiations with acceptable suppliers. The suppliers would be asked for their help in determining how to use their system. I/C firms may insist that the DBMS and other packages come from an independent supplier of software.

This practice allows for a cooperatively developed proposal to be put before management; one in which each party to the proposal knows his role. More than one firm may be requested to make such a proposal. Outside consultants who are familiar with the various aspects of such a proposal may be used "to keep everybody honest." Such an evaluation may take nine months to a year, but significant preliminary design on how to use the to-be-supplied hardware/software system would have been done.

In the Federal Government practice, whose major procurement dictum is maintenance of competition, the selection of hardware is predicated on a functional specification. For systems to be implemented in-house implies a hardware functional specification to permit various subsystems of the hardware to come from different vendors. Specific details of support software cannot be specified if such specification gives special advantages to only one vendor or, alternately, eliminates too many competitors.

At best, this leads to a watered-down software specification that may not be strong enough to support the intent of the internal Federal Government user. At worst, this leads to a strong version of the specification such that no one can supply as an off-the-shelf software system. Testing such a system will take place long after selection. Systems lacking extensive field use are notorious for being unstable and poorly matched to the job at hand. Examples of such systems are the WWMCCS and the ill fated ALS. It also means that non-vendor developed support software is seldom bid by the equipment manufacturer.

In terms of time, this system of procurement produces a one or two year procurement process which requires a benchmark testing period oriented to some standard language. Such benchmarks cannot take special advantage of a vendor-specific operating systems or DBMS.

After the hardware is selected, the internal software group must do a preliminary detailed design, using the specific hardware/software selected in order to determine whether the original requirements can be met. At best, this leads to an additional one to two year delay over industrial practice. At worst, it leads to a squabbling, contentious confrontation between two groups with even a possibility of law suits, threats, and ultimate abandonment of the original objectives.

Buying high technology from an adversary point of view rather than a mutually cooperating point of view leads inevitably to extensive delays and increased costs--if not to outright failure.

Note also that during the lag in the Federal Government procurement cycle, several things occur. The problem changes in scope. As noted in the keynote address, industrial growth in capacity required each year (stated not in dollars but in computation) grows 10-20 percent per year. The user gets discouraged and cancels; or worse, is given two more years to extract new promises from the internal ADP manager. In addition, while waiting for hardware/software selection, the internal ADP team must: 1) Bide its time; 2) gamble on the winner and proceed concurrent with procurement; 3) disband; or 4) assist the user in making larger plans. The first is wasteful, the second is risky, the third introduces delay and confusion after selection and the fourth leads to extensive over-commitment of both the to-be-acquired hardware and the internal manpower.

Thus, the panel found significant differences between I/C and Federal practice. While the differences may have much larger implications they also increase hardware costs for the Federal Government when buying a DBMS.

A way to avoid some of the difficulties described is for the federal agency to create a truly functional specification in the user's terms and to ask for a competitive turn-key total system. This will eliminate third party sources, etc., since the total system is to be procured and supplied by a single source. However, upgrading the system to handle additional growth requirements would require, at best, an interim upgrade and, at worst, a totally new procurement process. This would be anathema to a concern with a profit motive. Simply stated,

cost-conscious organizations would not consider the Federal Government's methods to be good business practice.

4.3.2 The First DBMS Installation. When installing a DBMS for the first time, management, user and service supplier must "get its feet wet" in a quick and successful project. This argues for phasing implementations into small packages deliverable in, at the most, four to six months. This allows each group to develop confidence, give feedback, adjust its plans and expectations to match better the tasks at hand and to feel, in general, comfortable with the changes.

Installing the first DBMS will result in changes to lines of authority and responsibility, some real and some apparent. Do not let the new data base administrators be stymied by the old war-lords. Fiefdoms exist and must be continued but the data base administrator is responsible to the organization as a whole and needs the necessary authority and management support. Of course, this new operating mode will require new responsibilities and corporate approval before old negotiated responsibilities can be discarded.

One way to get assistance in promoting acceptance of the data base administrator is to keep the communication lines open. This enables all to observe the changing environment, to get frequent statements of management support for the data base administrator, and to reaffirm the resolve to continue and complete the desired conversion.

It may come as a surprise to old hands in the business that conversion to data base environment from either a current file system or a manual system will each require a fault tolerant or "forgiving" mode for data base generation and update. Even from automated files, there may well be illegal values, voids, etc. in the file. This could cause a significant number of errors to be found by the edits when building the new data base, especially if internal DBMS pointers are value dependent.

Your users will need help. Do not "over automate" on the initial change from manual files. Use the current data in its current forms to meet today's needs. Get it into the data base and clean it up as the data is needed for new applications. Cleanup can be a significant and unplanned-for activity. Accept the dirty data and give some service of value. Do not take the position that it's their fault that the data base cannot be generated. Help them.

For those converting already automated systems, keep a parallel backup for each full system conversion until the new system is solid. This may be three to six months.

Try scanning a file for range of values on an item: illegal or nonsense values are frequently present.

4.3.3 Desired Standards.

The panel regreted the lack of common terminology in the data base management arena; indeed it appears that every new development of any size brings with it an opportunity to create a plethora of new names and verbs to distinguish minor variations. The underlying fundamental concepts should be given standard terminology and variants clearly explained and justified.

Conversion tasks would become easier if each data management system had the same functions available, or possibly a basic subset of some superset. Common functions with common definitions would create the same result for the user, even though the implementations varied. This may be both unachievable and undesirable, given the several differing fundamental file forms. However, it may be possible within classes of data base systems.

A third useful area for standardization is a micro-language (atomic verbs) in which the functions (commands) of a data management system can be described so that the detailed specific actions of each command are obvious. This would make definitions more clear and precise. It would facilitate comparing DBMS's and it would facilitate re-implementation of a DBMS or emulating one DBMS with another, an action which might be needed to facilitate changing hardware and DBMS.

4.3.4 Desired Technology.

The panel saw the need for tools to assist conversion from one DBMS to another. Unfortunately, they might have to be dependent on a particular situation but converting one DBMS file to another should be possible without going back to transaction mode or card image mode, the most prevalent way of generating a data base.

Another technology development needed is the creation of a model-independent data description that could be automatically mapped into any Data Description Language.

4.4.1 Introduction. The rapidly changing field of data automation ensures that opportunities will always be available for transition of application software. This transition may involve evolving from a non-DBMS environment to an exclusively DBMS arena, transitioning from one hardware environment to another through a common DBMS, transitioning from one DBMS to another on either existing or new hardware, or entering the distributed processing area. This section will attempt to explore the various aspects of these transitions and the role which DBMS and other tools may play in achieving the transition goal. Extensive use will be made of the effects of transfer from one DBMS to another. The specific effects which will be discussed include transfer of people, transfer of data, transfer of capabilities, and transfer of procedures.

4.4.2 Changing From Non-DBMS To DBMS.

Perhaps the most traumatic of conversions involves the initial move of an organization into the DBMS world. For the traditional application programmer, the DBMS looms as not only an unknown world but also as a threat to job security. The first task then is to provide the threatened programmers with the proper training, not only on the specific DBMS to be used, but also on the advantages and disadvantages of DBMS as a whole. Of great utility in the transfer of people is on-site vendor provided training. Ground work for the move, however, should initially be laid through commercially available courses dealing with data base systems independent of a vendor implementation. Initial training should begin either before or during the DBMS procurement cycle so that the present personnel will see the DBMS for its utility and not as a threat. Furthermore, thorough conceptual training will ensure greater utility from the DBMS vendor's on-site training. Care must be taken to ensure that the vendor's course includes a practical exercise to demonstrate the advantages of DBMS, specifically in the areas of interfaces, DDL, and DML. The wise planner will quickly recognize that the most important resource for training people is time. If possible, time should be set aside following vendor training for application programmers to experiment with the procured system before requiring production programming. This exercise may involve design and programming for data conversion needs.

The transfer of data from independent files to DBMS control can require extensive resources. The process is usually a download by stand alone application programs then upload through DBMS dependent application programs. Although the concept sounds easy, the volume of data as well as variety of storage formats, devices, and locations can, and do, drive the cost of transition to very high levels. The solution to this problem lies mostly in the continued efforts of managers to keep current their documentation on files and their uses. This documentation may take the form of written narratives and file format charts but perhaps the most effective tool is the data element dictionary/directory (DD/D). This automated system can provide managers with current information as to the type, size, location and usage of data. Although used most extensively in coordination with DBMS, the DD/D can ensure that managers and those responsible for the inclusion of data in a data base are provided a complete view of the agency data as a whole as well as usage of that data by application programs. In addition to these tools, current research by Jim Fry of the University of Michigan and Arie Shoshani of SDC [See the Conversion Technology panel report.] in data translation and conversion promise future tools for describing source data files and providing translation to some target format. Emphasis should also be placed on determining a common interchange form for the conversion of data between hardware types.

Moving from a non-DBMS to a DBMS environment involves, in most cases, a matching of like functions in many independent application programs to shared functions in the DBMS. Consequently, this movement involves removing code from the application programs and replacing it with code to interface with the DBMS. The bulk of application dependent code should remain the same. Capabilities such as backup/recovery and physical data structuring will be moved as a whole to the DBMS. In the DBMS environment, no capabilities are lost to the application programs--only gained.

No tools are currently available to aid in this transfer of capabilities, but all DBMS share a common repertoire of functions to include facilities for data creation, update, and deletion. These capabilities, however, vary depending upon the underlying conceptual model of the DBMS. Standardization of DBMS capabilities and syntax could allow for a more complete predesign effort through knowledge of system features.

As mentioned earlier, conversion forces management to take a much tighter control by establishment of a centralized function known as Data Base Administration (DBA). Within the DBA function, procedures are further established to deal with data integrity, accessibility, confidentiality, and miscellaneous control. The primary tool of the DBA is "clout" or having a position in the organization with the authority and responsibility to ensure the compliance with established procedures. The DD/D is also used by the DBA to aid in the design, implementation, and maintenance of the data base.

4.4.3 Changing From One DBMS To Another. Perhaps the least explored area of conversion and need tools involves the migration of applications from one DBMS to another. This conversion process is second in impact on users only to initial entry into a DBMS environment. The training of personnel on the new DBMS is simplified somewhat by a previous exposure to the "DBMS way of doing things." The application programmer, however, will be required to view the data with which he works in a new manner, especially if the move is from one conceptual model to another; e.g., from a network to a hierarchically oriented DBMS or vice-versa. Training will be the primary tool in conversion, including both vendor and follow-on in-house training sessions and exercises. For conversion between certain DBMS (CODASYL-like systems, for example), automated tools may be used for direct syntax transformation in both DDL and application programs. Conversion, however, between systems which do not share a similar syntax or mode of interface, e.g., TOTAL to IMS, IMS to DMS-1100, will require extensive training to learn the new Data Manipulation Language (DML), DDL, and interfaces.

In migrating data to the new environment, no new problems will be encountered which have not previously been discussed. The download programs should, however, already be developed for such purposes as data base archiving (dumping).

The transfer of capabilities provides the greatest impact on the conversion between dissimilar DBMS. DDL presents the first difficulty in conversion. In order to implement the new data base, some description of the new data base structure must be developed in the target DDL. Automated translation tools can be developed if the source DDL is extensive. Manual translations, however, seem more appropriate due to various changes in the way in which DDL may be interpreted; e.g., DATA SET in DMS-II is not the same as DATA SET in IMS.

A DML, as mentioned earlier, offers the most advantages to automated translation when the interface syntax is fixed (CODASYL's DML). When the interface is purely through dynamically built character strings (IMS, TOTAL) automated conversion can only replace CALL statements of one format to CALL statements of another. The DML functions, although common (GET, PUT, CREATE, DELETE, UPDATE), also vary in interpretation. An example: FIND in a CODASYL-like DBMS only locates a record as stated in the record selection expression and a subsequent GET must be issued to load the data into the issuing program's work area. A GU (get unique) in IMS not only accesses a record but also requires the Segment Selection Argument (SSA) to find the appropriate record. Therefore, the capability of a single DML command may not be replaceable by another single command of the new DBMS. Conversely, several commands which span both DDL and DML (record selection expression/set selection) in a CODASYL-like DBMS may be replaceable by one command (GU W/SSA). Other commands of the source DML may not be duplicated in the DML of the target DBMS. Apart from the DDL and DML capabilities, more basic differences may exist.

Shared functions which DBMS supply are often quite different. Where one DBMS may supply item level locks, another may make only record or data base locks available. Access controls, recovery, and auditing are just a few of the support capabilities which vary between DBMS, even those which implement the CODASYL specifications. The change in capabilities, therefore, is indeed the most evident conversion problem in migrating applications between dissimilar DBMS.

The transfer of procedures will usually be transparent to the application programmer but will impact heavily upon the DBA. Because of the differences in capabilities and architecture of DBMS, the DBA will be forced to adapt fastest to the changing environment. The DBA must immediately recognize differences in the operation of the new DBMS and take steps to avoid turmoil. Procedures must be changed as little as possible in external appearance. The application programmer must interface with the DBA in the same manner as before. The DBA, however, must determine and apply new procedures to ensure maximum system efficiency based on the information passed across this interface. The DBA's primary tool in this environment will then be training and any available consulting services.

4.4.4 Changing Hardware Environment. The most frequent motivation for conversion is caused by an impending upgrade in hardware from one manufacturer's equipment to another or between non-software compatible lines of the same vendor. Within this environment, change can extend from simply migrating applications from a DBMS on the current machine to the same DBMS on the target hardware (TOTAL on IBM to TOTAL on CDC), through changing DBMS between machines (IMS on IBM to DMS on UNIVAC 1100), and, finally, to initial implementation of a DBMS environment on the new computer. Two of these situations (DBMS1 to DBMS2, non-DBMS to DBMS) have been previously discussed. The additional impact of changing hardware, however, adds a new dimension to the training problem. Migration of data also is a bit more difficult in that some form must be established in which to download data for subsequent upload on the target machine. Transfer of both capabilities and procedures apparently are not impacted or complicated by this transition.

The additional transition of DBMS on the current hardware to the same DBMS on the target hardware begins to show the utility of standardizing both interfaces (end-user-facilities) and capabilities. Whether the DBMS in question is CODASYL-like or not, costs are significantly reduced when syntax and capabilities remain constant across hardware types. Examples of systems which provide these transition opportunities include TOTAL, SYSTEM 2000 (Non-CODASYL) and IDMS (CODASYL-like). Applications which utilize these systems can concentrate on more operating system dependent transitions (JCL1 to JCL2) because of the relatively small changes required in DBMS oriented syntax. Because of the vendor enforced (or de facto) DBMS standardization across machines, automated tools can greatly aid transition and may be as simple as filtering source programs and recompilation.

4.4.5 Centralized Non-DBMS--distributed DBMS. With cheaper system communication costs and better cost/performance ratios of the minicomputer (some supporting virtual memories in the millions of characters and most, if not all, offering large-scale peripheral disk storage), the use of data base software can be relocated from centralized large-host environments to mini-based decentralized arrangements of data processing facilities. Conversion from the traditional centralized mode to distributed data base processing is fraught with both technical and management perils. The mini-based DBMS generally does not offer the diversity of facilities available with data base systems centrally resident in larger hosts and, in fact, may constrain the design of the application systems planned for dispersal. Replicating the application software, data base structure (DDL), and data base software may require a severe

reorientation of existing ADP management philosophy. Management and control of multi-site operations, ADP standards, application development and data base administration will be made more difficult. Conversion of data previously centralized requires that some intermediate data form be created for downloading appropriate subsets of the old file(s) and subsequent uploading on the distributed hardware. Standardized interfaces (DDL, DML and end-user facilities) and common DBMS capabilities will reduce the management problems encountered during the design, development, conversion and operation of distributed data base implementations (even if data base management software and mini-computer hardware differ from site to site).

4.4.6 Centralized DBMS--distributed DBMS. The problems of conversion from a centralized mode of DBMS operation to distributed data base processing are somewhat eased by the presence of experienced and knowledgeable personnel at the beginning of the conversion, but are still of concern. Aside from the difficulties associated with the migration of existing application code (dependent upon the existing business functions to be dispersed, degree of transferability of the programming languages presently in use, limits on size of load modules, etc.), further problems surface if dissimilar data base logical structures (network, hierarchical relational, etc.) are encountered.

Conversion to a distributed data base structure that is similar to the existing centralized structure should be an easier task. DDL differences may exist, however, between the existing and target data base systems as well as the obvious job control language differences. The target (distributed) data base software may not offer the facilities used (or planned for use) in the existing centralized applications, and support of user business processes may thus be reduced. The problems noted in the conversion of centralized non-data base applications to a decentralized data base mode remain unchanged for the migration from centralized to decentralized data base modes.

4.5 GUIDELINES FOR YOUR FUTURE CONVERSIONS

The panel concluded that the single most important guideline to offer a group about to embark on a system conversion was to use their best management techniques. It is a technically difficult job, like most large software system developments, and one must apply all well-known software development methods.

4.5.1 General Guidelines. The discussion of the various conversion scenarios revealed the fact that several management practices had been utilized in all of the successful conversions which, though common across all data processing environments, were of particular relevance in the data base environment where problems and errors in establishing policy proliferate across all applications utilizing the DBMS.

4.5.2 Important Considerations.

- Analyze all possible file organizations. Do not assume that the most efficient way is going to be a data base. Some applications can perform better using sequential files such as tapes or other access methods such as Index Sequential.

- Analyze the final stages of the conversion. How will you convert to production? Will it be (or can it be) converted module by module, transaction by transaction, or will it be necessary to "push the button" and convert to production all at once?

- Develop a representative model of the company's business for design purposes. This model should provide the basis for the design of the data base which will be in production.

- Determine the degree of security and integrity required for the operation of the data base. Who will have access to certain information? How will unauthorized use be prevented? How will you recover or reconstruct the data base in the event of a software or hardware failure?

- Determine the space requirement for the production data base. Determine if all the data on the file is necessary and is being used. Check for redundant data. (It is not always bad to have redundant data, especially if it will improve retrieval time.)

- Build a small version of the data base and test for deficiencies in the design or in the modules. The testing must be thorough and cover all facets of the company's business. Users' involvement in this stage should be heavy but should not dictate how the data base is to be designed or what access methods must be used.

- Determine whether a Data Base Administrator is needed for your organization. The DBA will be responsible for the security and integrity of the data base (including recovery and backup) and act as an agent between the User's group and the data processing group.

- Determine the type of support you will need from the vendor. Will classes be given on data base technology? Will the vendor be readily available when a problem arises? Contact various users in your area. Attend a local users group meeting and find out the types of problems and solutions that the group has come up with, especially talk to users in the same type of business. Valuable information can be obtained and you will not re-invent the wheel.

- Determine how eager upper management is to undertake the task of converting. If they support you, the task will be easier and more efficient.

- Look into the future. Determine whether the present design of the data base is the most efficient one in case new applications are introduced. Do not be afraid to redesign if it proves to be more efficient. Provide for a purge of unneeded data. Reorganize the data bases on a regular basis to reclaim unused space or to compact the data. One very large data base may not be feasible.

- Determine whether to use standard vendor supplied software or develop you own.

- Once a decision has been made to convert to a DBMS, check and re-check for contract to be signed. Have a contract attorney study the agreement.

4.5.3 Tight Control. In successful conversion efforts, special emphasis must be placed on establishing procedures and policies and insuring that they are followed. Examples of these procedures and policies include internal standards, data base administration, and explicit (precisely defined) goals.

4.5.4 Precise Planning/pre-planning. The probability of a conversion succeeding is directly proportional to the degree of preparation. Preplanning efforts involve:

establishing current baselines and identifying perceived inadequacies

- defining requirements or specifying what are the desires of the conversion

- enumerating as many alternatives as practical and determining their appropriateness in the target environment

- establishing feasibility in terms of costs, people, and time

- describing milestones and benefits

- receiving and documenting a management commitment to a specific alternative

Planning efforts must begin immediately upon management commitment and include:

- organizing for the project

- defining support software requirements

- evaluating available systems (packages)

- describing selection criteria

- selecting and procuring software

- building a well-defined implementation plan including staffing, training, detailed design, and implementation strategy

- seeking final plan coordination and approval

4.5.5 Important Actions. The panel recommended several actions:

- Do Establish Review Points. When describing a plan, care must be taken to ensure flexibility in order to support changing/overlooked requirements. Points must be scheduled at which time the plan is reviewed and updated as required. Decisions must also be made as to the appropriateness of proceeding with development.

- Do Involve End Users. The data processing department must be extremely careful not to ignore user input to the development process. No one knows the functional requirements which must be met better than the end user.

- Do Keep Scope Reasonable. The scope of the project must be dictated by practicality. Attempt only as much as is attainable within well defined time frames and technology. Once the scope is defined, stay within its bounds.

- Do Not Stifle Prototyping. Modeling is perhaps the best method available for trying out ideas. Encourage validation of concepts through prototyping.

- Do Shift Responsibility to Where the Expertise Exists. Always ensure that responsibility for achieving goals is placed where the ability to both understand and accomplish them exists.

- Do Phase the Implementation. Within an overall context, ensure that total system conversion is planned and executed in attainable increments.

- Be Wary of Entanglements. System conversion planners must always review commitments with the goal of avoiding crippling dependence upon manufacturers, proprietary packages, and maintenance agreements. Recognize Costs/Benefits of New Technology. Quantify both the advantages and disadvantages of being the first to use new technology.

- Do get adequate management commitment. Make sure management knows beforehand what resources and time periods are required, what has been available, and what progess, if any, has been made. Do not let them walk away from it. They are key players.

- Do initially select an important but tractable portion of the ultimate and have some results in four to six months. This forces a early and realistic review. One also hopes it demonstrates the new capabilities, emphasizes the ability of the implementors to handle the new technology, and provides better insights into both the technical and operational (people oriented) problems that the organization will face.

- Do introduce your technical staff to the new technology. First, when you are studying whether to go data base but, also, immediately before and during the initial implementation. Sometimes six months to two years pass from initial study to start of implementation. Further, training for the initial decision process is usually "book learning"

and quite stale by implementation time.

- Do orient and educate your users. Next to management, this group is the key to success. They must feel comfortable with what you say. That means they should have a role in deciding what they get and when. In addition, they must prepare themselves for the new modes of operation.

- Do keep the user group on your management review team. They can keep you out of trouble. They can change their priorities and needs to fit your capabilities to deliver. They can help you resist pressures for early and extra delivery since they, too, want and need a quality product. By keeping them in the loop, you risk less chance of surprising them and evoking resistance on delivery.

- Do set up a central "data base control group." Useful in any big system or series of related systems, it becomes essential in a data base oriented organization. Such a group may have several levels with differing managers but these groups must be coordinated. There is a policy level, an implementation level, and an operational level. The implemenation group does most of the administrative and design work involved in bringing into being and providing for the efficient structure of the data base(s). The operational level is responsible for the operational integrity of the data base and must take the actions for restart/recovery in addition to any periodic initialization/dumping or consolidation of the actual data base.

- Do implement a data dictionary/directory. As a key item common to all systems, it is needed to make the data base a corporate rather than a private entity. Because the data is to be shared across groups who are not responsible for the data, it is essential that the "meaning" of each data item be documented. A data item, besides having a name, a form, and values chosen from some well documented set, has several meanings. One is the English description; another is an operational description of how the value is determined; another is how the data is commonly used and by whom. All of these must be known to avoid confusion and to avoid having data misused.

- Do ask for and accept help from the vendor of the data base system. If you are also changing hardware at the same time, put pressure and some of the risk on that vendor. Good advice must be available during your learning period. Consultants are useful, not only initially, but also during design and implementation time.

4.6 REPRISE

The most significant finding: the ingredients for success are management commitment and discipline, skilled people, clear roles, and lots of cooperative conversation between the parties involved. Not a new discovery but still significant--and more essential than ever at this stage in DBMS evolution.

4.7 ANNEX: CONVERSION EXPERIENCES

4.7.1 Conversion: File To DBMS. The three user experiences which follow illustrate the scenario of converting from a file system environment to a data base environment without a change in hardware resources.

Conversion Experience - A.

Goals. Studies were conducted to analyze the problems of the file environment. Problems identified by the functional user and data processing departments included prohibitive maintenance and enhancement costs, slow implementation of additional user requirements, heavy data and processing redundancies, and lack of data integrity.

Summary of actions. The problems described in the previous section led to the data base decision. Following this decision, a consulting firm prepared a data base conversion plan to execute those tasks in the plan leading up to but not including the conversion of data or application systems. A directory of steps and tasks in the data base system plan that the consulting firm prepared follows.

Step 1: Evaluate the Present Information System
- Task 1.1 Information Flow Analysis
- Task 1.2 Current ADP Analysis
- Task 1.3 Current Reports Analysis
- Task 1.4 Status of Present Information System for Data Base
- Task 1.5 Draft of Data Base Administration Responsibilities

Step 2: Define Data Base Requirements
Task 2.1 Service Analysis Development by Report
Task 2.2 Service Analysis Development by Function
Task 2.3 Data Dictionary Development

Step 3: Develop Initial Data Base Architecture
Task 3.1 Distribute Data Dictionary Elements
Task 3.2 Distribution Optimization
Task 3.3 Architecture Description

Step 4: DBMS Package Evaluation and Recommendation
Task 4.1 Architecture Mapping
Task 4.2 Support Features
Task 4.3 Secondary Features
Task 4.4 Recommendation

Step 5: Application Redesign
Task 5.1 Design and Program Documentation
Task 5.2 Structured Programming Analysis
Task 5.3 Implementation Controls

Step 6: Configuration Analysis and Evaluation
Task 6.1 Current Load Analysis
Task 6.2 Future Load Analysis
Task 6.3 Simulation Modeling
Task 6.4 Simulation Analysis

Step 7: Implementation and Conversion Plan
Task 7.1 DBMS Installation Planning
Task 7.2 Data Conversion Planning
Task 7.3 Application Conversion Planning

Step 8: Personnel Development and Training
Task 8.1 Assess Available Talent Levels
Task 8.2 Develop a Formal Training Plan

During these tasks the consulting firm provided project direction and DBMS expertise. Company personnel assigned to the project received practical training while preparing for planned conversion of application software and data files.

Some points should be noted about the tasks prepared for the data base system plan:

- The evaluation of the results from tasks 1.1 through 1.4 was used to support the data base decision.

- The data base administration function was established following a draft of responsibilities prepared in task 1.5.

- The initial data base architecture described in task 3.3 was package independent.

- The DBMS package recommended in task 4.4 was influenced by constraints imposed on the evaluation process. The influences included the need to interface with teleprocessing and the ability to install DBMS with an initial or pilot application system conversion in just four months.

- The application conversion plans in task 7.3 identified functional redesign requirements resulting from the service analysis conducted in task 2.2 to correct deficiencies in existing software and to benefit from the facilities offered by the DBMS package.

At this time the recommended DBMS is installed and the conversion and implementation of the pilot application system and its data files to data base is completed. The conversion effort was successful but not without its problems. The functional user organizations feel the benefits of DBMS in the areas of performance, data integrity and responsiveness on the part of the data processing department to change requests. However, further implementations to DBMS are suspended due to a redefinition of priorities outside the data base system plan.

Conclusions. Factors contributing to the success of that conversion or to the problems that occurred were:

- Management demonstrated commitment by the funds made available for consultant services, by the investment in preplanning and planning tasks, and by the establishment of a DBA function. This commitment was instrumental in the responsiveness and cooperation that was sorely needed from data processing personnel and users alike when the data base project was undertaken.

- The establishment of a DBA function in the data processing organization invariably creates a political struggle between the DBA, the Manager of Systems and Programming, and the Manager of Operations. The use of consultant services minimized the political problem; at least during the preplanning and planning phases. However, the impact or trauma is felt most during the pilot application system conversion process. The lack of adequate management support to the DBA function as demonstrated by its placement within the data processing organization and the limited personnel

resources only aggravated the political problem. As such, the establishment of the DBA function was less than effective. The influence of such political struggles on the suspension of further data base conversion activities is difficult to determine.

- A major problem in converting from a non-data base environment to data base is that during the period when data base expertise is critically needed, you are least able to provide it. This problem was eliminated with the expertise provided by consultants.

- The placement of the DBA within the data processing organization and the limitation of its responsibilities contributed to the lack of support at the right level of management for continuation of the data base project. This was very much in evidence when staff meetings were conducted to prioritize data processing activities.

- The benefits realized by the conversion and implementation of the pilot application system resulted from the redesign activities. However, the cost/benefit analysis was instrumental in identifying a pilot system that had low risks, early benefits, and visibility. These three factors alone may very well save the future of the data base project in the context of further implementations of data base. Once a data base conversion effort is suspended, it normally takes support from the user community to reactivate it.

- Problems encountered during the conversion process were due, for the most part, to inadequate training of people in the usage of the DBMS and in the interpretation of its diagnostics. The amount of training planned was reduced in an effort to compensate for an insufficient number of people made available to convert the pilot system.

Conversion Experience - B.

Goals. The experience involved choosing and using a DBMS to support processing of solar hardware system data. The solar hardware data resided in file format, yet structural relationships existing between and among data records could not be supported in any reasonable manner in the file oriented environment. In addition, many of the user requirements were not adequately defined and there was considerable worry about whether the known requirements

could ever be met.

Summary of actions. A group was gathered to define the users' information processing requirements and to determine whether a DBMS solution could be provided within a reasonable time at reasonable costs. Initially, all combinations of hardware and software solutions were considered because the data processing center within the organization did not have a DBMS. As time progressed, the users' requirements became clearer and time-sharing service solutions--the desired mechanism for achieving a hardware change -- were dismissed because of control and cost factors. The conclusion was to provide a DBMS for the existing hardware.

Although unanswered questions remained regarding the feasibility of this approach, many desirable aspects were, on the other hand, apparent. The DBMS selected was available in a bundled form from the hardware vendor--no additional costs--and the software could be delivered in a timely manner. In addition, the selected DBMS supported the required structural relationships with its CODASYL orientation.

The best understanding of what output requirements the DBMS should provide was used to design the data base and implement application programs. All of the requirements were met in a reasonable time within reasonable costs.

Conclusions. Critical points in the success of this DBMS conversion experience were:

- DBMS expertise was available from the beginning--a most important time in any DBMS environment.
- The users participated in defining the information processing requirements.
- Expertise was available in matching information processing requirements to available DBMS capabilities.
- The vendor provided adequate training and sufficient time to gain experience after the training was completed. This provided the necessary insights into the actual capabilities of the chosen DBMS.
- Portions of the output requirements were sufficiently well documented and were of reasonable magnitude to demonstrate successful operation of the DBMS demonstration.

Conversion Experience - C.

Goals.

- To convert a large number of antiquated files to the state-of-the-art technology.
- To provide for a more efficient way of retrieving and updating data.
- To eliminate redundancy through a centralized data bank.
- To eliminate erroneous information through data integrity.
- To provide better management of information.
- To provide for more system security.

Summary of actions. Only one alternative was available to convert existing files to a DBMS and at the time only one DBMS could meet the requirements set by upper management. A hierarchically structured data model DBMS used exclusively with its own higher level language interface was chosen.

The first conversion had to execute the load procedures eight times until all known errors were eliminated. Each load took over 100 hours to run with no checkpoints. Fortunately no crashes occurred during any of the runs. The load process, in itself, was not a major problem; however, some erroneous data present in the old file was input to data bases. The users were encouraged to be involved in checking and re-checking the new system. At times, the users requested changes which, to them, seemed minute, but which required large programming and system changes. Some users wanted to have more information in the data base (such as segment's last date of change) while others were concerned with space and efficiency. We provided a backup system which we could fall back to in case the conversion failed. Meetings with users continued for six months until repeated testings satisfied us that we had reached a satisfactory level of accuracy.

Another problem resulted from personnel assignments. While one group was loading the data base, another was testing. The group that loaded the data base did not have as good an understanding of content as the testing group. Consequently erroneous data was loaded. If both groups had worked more closely, most of these problems would have been

resolved.

Conclusions. The following points are characteristic of successful conversions in the described scenario:

1. Management commitment ensures sufficient resources and end user support of the conversion effort.

2. DBMS expertise must be available from the beginning of the conversion effort.

3. Adequate preplanning and planning activities are instrumental in the conversion to data base.

4. The end users must participate in defining information processing requirements.

5. The DBA support must be a strong and continuing commitment.

6. In the DBMS package evaluation, expertise must be available to match the information processing requirements to the capabilities of available DBMS packages.

7. Adequate and timely training must be provided and sufficient hands-on experience must be gained prior to any conversion of application software.

8. A conservative staging plan, one which selects an application system for initial implementation that offers low risks, high benefits and visibility, must be developed to insure continuing resource support.

9. The DBMS conversion must be thoroughly and adequately tested prior to the production cycle. A fall-back procedure should be established in the event that the conversion is not successful.

4.7.2 Conversion: Manual Environment To DBMS. This scenario represents conversions in which data is maintained and used manually in the source (old) environment and it is desired to convert the data and functions directly to a DBMS without first introducing a non-DBMS file system.

Conversion Experience - A.

Goals. The goal of this conversion was to mechanize functions performed manually by separate operating departments using the same input documents. The input documents triggered each of these departments to perform its

own functions using its individual, manually-maintained data records. The flow of the input documents went from one department to another. The flow was sequential in nature since one department had to complete its function before the next could begin. Each department independently maintained its own data records. The records in each department contained some common data components.

This total process had never been mechanized before because of the complexity of the operations involved and the lack of data structures to represent the interrelations of the data and functions. DBMS technology made mechanized processing and a common data storage place feasible.

The expections of mechanization included:

1. Better performance of end-user functions through more accurate records and increased capability of machine over human operation.

2. More efficient performance of functions through elimination of duplicate record keeping and mechanization.

Summary of actions. The project selected a network DBMS including full communications and transaction management capability. Interactive terminals were designated for end user locations as well as an interface developed with a front-end system controlling the flow of the inputs to the system and the distribution of the outputs to the appropriate user departments and to other mechanized systems.

The conversion from a manual environment to the full DBMS has been completed successfully at one site and the DBMS has been operational for several years. During this time, more sites have been converted to the system. Interfaces between this system and other DBMS systems are being developed. Since these interfaces are not yet operational, conclusive findings cannot be stated.

Conclusions. The task of interfacing two DBMS should not be underestimated, especially if the systems use different software or hardware. For example, plenty of time should be allocated in developing the interface simply to work out the kinks of reading data created by another DBMS in a language with a different bit structure and character set.

When an interface is first developed between two systems, time must also be allotted to resolving differences in how one system identifies or names what are thought to be common data elements. Both systems may process the same

widgets, but one may be concerned with inventory and the other with maintenance or repair. Both systems can be involved with some of the same widgets, but the way they identify or represent them may be different. These differences may not be apparent or may seem trivial until an actual mechanized interface is attempted.

Merely because there could be a mechanized link between two systems does not mean there should be such a link. It may be far more economical to use a manual interface, especially for low volume interactions or in cases where the viewpoints of the two systems differ greatly.

Environments with multiple DMBS are very frustrating to users if these DBMS each use different input devices with different sign-on procedures and modes of interaction.

Coordination and planning between the DBMS in development are needed to prevent proliferation of different terminal equipment at the same user locations.

Conversion Experience - B.

Goals. The goal of the project was to automate a manual document retrieval system.

Summary of actions. In this project, management greatly "simplified" the decision process by directing conversion to a custom-built document retrieval system which would be converted to our local hardware (i.e., converted from one hardware manufacturer to another). Apparently, the reason for this decision was that a copy of the software could be obtained free. A software conversion (COBOL-to-COBOL) had to be done before the local document retrieval application could be automated with the software. Then, the software had to be augmented (new code) to add needed functions, including: (a) more extensive editing of input data; (b) more user-oriented processing of retrieval requests; and (c) redesign of outputs.

After the software conversion started, analysis made it apparent that the capture of the manual data in a machine-readable form would require about twelve staff/years. Also, on-going entry of new documents and servicing users would have required a permanent staff of 4-5 people. Attempts to convince management of these needs were unsuccessful.

The events that followed were:

- A successful software conversion followed by augmentation of the converted software.

- Design of procedures and entry forms, and identification of resources needed to capture the manual files.

- Shelving of the project because resources were not available to either contract the data capture or to staff it in house.

Conclusions. While not a conversion to a true DBMS, this experience illustrates, by their absence, several points necessary for DBMS conversions.

- Importance of planning and analysis before any decisions are "cast in concrete."

- In this case, the software was decided upon before analysis. And the software chosen was totally inappropriate for these reasons: (1) It was not running on the same brand of hardware as the local system, so it had to be converted. (2) It was custom-built and no support was available from the originator. (3) Software documentation was fragmentary and obsolete--we got source code, compiler listings, test data, and very little else that was useful. (4) The software did not provide adequate functions so it had to be augmented with new code.

- Importance of obtaining a commitment of resources sufficient for the entire project before starting any part of it. In this case, the decision-makers had no comprehension of what would be involved. Neither the one-time data capture nor the on-going support staff needed could be had when the time came to actually implement the retrieval system and service customers with it. Therefore, extensive conversion and analysis were wasted.

Recommendations.

The manual-to-DBMS conversion situation has two characteristics which make it unique among the Data Base conversion situations:

1. Data not already machine-readable

2. Users not accustomed to automated systems

Consequently, those contemplating such a conversion should take the following actions:

Develop implementation strategy. Purpose: Control cost and meet schedule.

- Limit scope. It is necessary to limit the scope of the conversion to something manageable and do-able. Limit objectives to high-volume, high-usage functions. Do not try to automate low-usage and/or exception cases. Stick to your initial limited scope. Do not be tempted to agree to ad hoc extensions of scope. Prioritize and save good ideas for future enhancements.

- Plan on phased implementation. Get something useful up soon. (Your limited scope--above--should have selected a useful small application.) Getting something into actual use soon will:

- Let the customer see benefits early and provide experience on the system
  a. Gives the project team some feedback early
  b. Buys the project team some credibility with users for later, larger projects

- Plan on re-implementing. The first limited-scope application(s) will very likely benefit from later re-implementation.

- Select initial installation/site carefully. Give yourself the best chance of success. If the system is planned for several sites, do the easiest one first.

Understand magnitude of data capture and data cleaning effort. Everyone, including top management, must understand that capturing the data and correcting it will be a 'large' effort. This effort is a significant part of overall project costs; it may be the largest.

It should be understood and budgeted for at the beginning. Sampling the manual data for accuracy and completeness is useful in estimating the resources needed for data capture and correction.

- Resolve political problems of ownership of data.

- Source. Where the data is kept by several organizational divisions, determine which one is the best source for the application you are converting.

- Discrepancies. Designate someone to be responsible for resolving discrepancies.

  - missing data elements
  - incorrect data elements
  - Obsolete data elements

These, of course, are familiar functions of the DBA. Such problems are, however, magnified in a situation where there has been no automation before.

Involve users early. Early end user involvement can:

- Get some support for the project team

- Give end users time to identify:

  - Changes to their work flow
  - Changes to staff requirements
  - Training requirements

Inconsistency tolerant systems. In going from a non-automated to an automated environment, higher incidences of errors or data discrepancies are likely to be found. This is especially true in initial installations because the new application software or system software may process the data incorrectly. The system should be designed to be defensive in all aspects--from program design to backup and recovery procedures. The system should be able to handle data discrepancies and inconsistencies gracefully. It should provide meaningful outputs in the face of data conditions it cannot process. It should provide easy to use tools to correct data problems that are found.

4.7.3 Conversion: Batch File System To a DBMS.

Conversion Experience - A (a Federal Government Agency).

Goals. The goals of this agency were:

- to replace outmoded hardware

- to implement centralized data management under a data base administrator

- to implement both new applications and redesigned systems in an integrated data base, on-line disk and telecommunications environment, under a data base management system, in order:

- to permit simple query capabilities

- to treat data such that:
  a. input multiple-use data only once
  b. have it accessible, locally and from distant cities, by terminal
  c. allow unstructured queries to be processed through the use of a query language without the necessity for programming

Summary of actions. As a Federal agency, the options were largely dictated by Federal procurement regulations. If there were no regulatory constraints, the alternatives would have been:

- to determine the features needed by the data base management system, evaluate existing DBMS's and buy a computer on which the most suitable one for their needs would run.

  This option was not allowed by the procurement regulations, because it would have resulted in a sole source procurement.

- To include in the request for proposal the DBMS required features. This option was not allowed because, in their case, the requirements would have unduly restricted competition.

- To procure the hardware under open competition, and procure the software separately.

However, regulations do exist and the agency had to select the final option which the procurement regulations had forced upon them.

Conclusions. The results, in terms of their goal of creating an integrated data base under a data base management system, were disastrous. The computer system that was procured was one on which a suitable DBMS did not exist. After two and a half years of effort to procure, through established DBMS software vendors, a system which could be made to function on this computer, either through conversion of an existing DBMS or the use of research and development software, the agency has not yet found a solution. Even after a contract is eventually awarded, they

anticipate a 12 to 18 month wait for delivery of the product.

Therefore, it is safe to say that their goal of moving onto new hardware and a DBMS has been so delayed by Federal procurement policy that it will take 3-1/2 to 4 years for them to recover.

Conversion Experience - B (a Federal Government Agency).

Goals. To convert three stand-alone systems with the following features:

- Input. Batch systems. Inputs manually transcribed from messages onto cards.
- Outputs. Outputs to other subsystems in the form of card images on tape that was hand carried, as well as printed reports.

The features of the new system were to be as follows:

- Inputs. On-line system. All error corrections processed and corrected interactively on a CRT terminal.
- Outputs. Printed reports.
  Ad hoc queries available through a DBMS query language.
  Outputs to other systems automatically generated and forwarded via a communications network.

Recommendations.

1. Conversion is usually a one-for-one affair. Redesign occurs when the requirements are changed. Conversion is usually an excuse for partial or complete redesign. A common mistake is to assume that for the price of a conversion effort, one can also redesign. Experience has shown that this is not the case. Unrealistic estimates of needed time and resources result.

2. The need for expertise in the new DBMS is greater at the beginning of a project than at any other time. Ironically, this is the time before your staff is retrained and when your expertise level is the lowest. Therefore, outside consultants must be brought in. These outside experts will have a great impact on your data base design.

3. The DBMS experts will, at an early stage, have to design the data base structure. The system design will then build upon this data base structure. The process of data base design and system design will be iterative with a series of changes until both are in sync.

4. The extreme pressure to get off old machines and operational on the new machines prevents sufficient time for thorough analysis and planning. And, as discussed above, if you are redesigning and not converting, the time pressure becomes greater.

5. Government procurement regulations do not recognize that a system is as dependent on the software needed as it is on the hardware.

6. The move from separate batch systems to an on-line integrated system, coupled with the need to learn how to use new hardware, require careful planning and extensive training in both new concepts and new techniques.

7. Moving into the centrally controlled DBA environment requires lead time in the systems development cycle for data dictionary development and implementation, and for data standardization to be done carefully.

8. Time required to accommodate recommendations five and six is seldom available when a move to new hardware is underway. The result is hasty and costly straight conversions of obsolete systems; therefore, the potential benefits of the new hardware are delayed or lost.

4.7.4 Conversion: DBMS-1 To DBMS-2.

Conversion Experience - A. This conversion illustrates the experience of moving from one DBMS to another, with a change in vendors.

Goals.

- To provide for the ability to update concurrently and retrieve information from the data base.

- To provide for a quick response time for inquiry purposes.

- To provide for the capability to have on-line inquiry even if the DBMS is in abort status or in recovery stages.

- To provide an efficient way to recover and backup the data base.

- To save money in the long run.

- To provide tools for security and integrity of the DBMS.

Summary of actions. At the outset several alternatives were considered:

- Continue with the present DBMS. This alternative implied maintaining duplicate data bases--one for inquiry purposes and one for updating purposes. The inquiry data base was at least one day behind the second one and at most a week behind the latter. In order to make the inquiry data base as current as possible, an image copy was made once each week of the updated base and copied to the inquiry data base.

- Update the DBMS using the same vendor.

- Convert to another DBMS using a different vendor. The new vendor agreed to convert the bulk of the system quickly and with the least amount of logic changes to existing modules. This vendor would meet most if not all of the goals.

Upper management decided to convert to a DBMS using a different vendor.

In order to meet the restriction of making no logic changes, the new vendor developed an Emulator to translate dynamically (at exeution time) calls in COBOL programs using the old vendor format without requiring manual re-coding. Only format changes were required and these were handled automatically via a standard vendor software package. The Emulator itself was developed by some of the best technical people available at the vendor site. It sounded and looked great--minimal programming changes and transparent to users. In fact, the programs looked as if they were written in the old vendor DBMS. The Emulator did, however, have certain drawbacks. The overhead incident to using the Emulator was extremely high since the calls were converted at execution time rather than at the source level.

Using the initial version of the Emulator would have resulted in failure to maintain the necessary production schedule. A task force was brought together to assess the matter and determine exactly how and where the system could be improved. The task force included site staff (three) and vendor staff (two persons).

The task force recommended changing some Emulated programs to execute under standard software and "looked into the Emulator internals" in an attempt to improve some of its functions. The changes yielded significant improvements; after the changes were effected, the regulator production schedule was met.

Another significant problem arose when the Emulator did not perform properly, e.g., a call to the data base was mishandled. On the rare occasions when a malfunction of this sort occurred, the following steps were taken to solve the problem:

- Disable the erroneous call
- Inform the vendor of the problem, providing complete documentation on the problem
- Wait for the solution
- Re-test the call
- Re-load the new version of the Emulator to the system library.

The prospect of losing technical support for the Emulator, in the event of personnel turnover among the vendor staff who designed and developed the system, was another potential problem. This situation did materialize, but the vendor was able to continue technical support with other individuals.

Conversion of the data bases themselves went extremely well. Standard utilities were used to unload the data bases. The tapes were moved physically to the new vendor's facilities and were converted via a standard vendor utility. The sub-files were loaded employing user written programs.

Since the on-line system had been installed on the new system well before the other applications systems, there was a need to ensure that the data base on the new vendor system was as current as the old. With smaller data bases, we unloaded daily from the old vendor and loaded daily on the new. With the exception of our largest data base, the other data bases were unloaded and loaded weekly or monthly. To

keep the largest sub-file current on the new system, we captured the data base changes on the old system daily and applied them to the new data base. These procedures lasted for approximately one year until our other applications systems were completely converted.

Conclusions. To summarize our findings:

- Differences between terminology associated with each system were found to be the initial barrier.

- Incremental transition was recommended as a means of graduated "phase-in" to the new DBMS.

- An intermediate software system was procured as the method of transitioning application code. DDL and DML functions appeared the same across the transition. Run-time interpretation of code through calls to interpretive-software was used to provide a mapping from original DBMS to native-made DBMS calls. Preliminary conversion of DDL-type facilities was provided at compile time with mapping structures being derived for use at interpretation time.

Recommendations.

1. Experience in interpretation and run-time mapping has shown that system overhead can be prohibitive. Problems encountered in execution require resolution by the developer of the interpretive software and can result in excessive response times. It is thereby seen to be more practical to utilize standard vendor software where possible. Conversion costs may be higher in the initial investment but will eventually be equaled and even surpassed by the overhead costs of the interpretive method.

2. The development of the interpretive software was handled by the vendor, with valuable insight of users. User involvement in the implementation of the interpretive software provided a more timely product with greater response to requirements.

3. The interpretive software, by nature, required processing support over and above that required for actual application processing. Direct transition could possibly have utilized to a greater degree the processing efficiency of the target DBMS. Again, this leads to the realization that should efficiency be a major consideration, direct transition may be more desirable.

4. In utilizing direct transition methods, vendors should be encouraged to develop standard methodologies for conversion processes in data description, data manipulation, and data base transfer.

5. Incremental transition insures that timely conversion can be established for applications. Prototyping efforts are provided through validation of earlier increments of the entire transition process.

6. Fall back positions must be established. The transitioned software must be completely validated before actual acceptance. The ability must always exist to turn-down transitioned software due to errors without degrading overall effectiveness of the data processing operations.

5. STANDARDS

Milt Bryce

CHAIRMAN

Biographical Sketch

Milt Bryce is President of M. Bryce & Associates, Inc., Cincinnati, Ohio, a company specializing in providing management consulting services in areas of systems design, data management, and project management and control. Mr. Bryce has been involved with the information systems field since 1951, working both for major corporations as an MIS Director and with Univac in product planning and systems programming. He has traveled extensively around the world conducting seminars in systems design and development and data management. He has been involved with standards development for the last fifteen years and has pioneered since the early sixties in structured information systems design and data management.

Participants

Robert Bemer
Don Branch
Jean Bryce
Elizabeth Fong
Al Gaboriault, Recorder
Anthony Klug
Henry C. Lefkovits
C. H. Rutledge
Philip Shaw
Jay P. Thomas
Ewart Willey
Jerry Winkler

5.1 INTRODUCTION

5.1.1 Objectives. The objective of the Standards Working Panel was to recommend standards that a manager should consider when converting data from present sources (manual, semi-automated, or automated) to a computer Data Base Management System (DBMS as it is commonly known). The Panel considered administrative guidelines as well as technical standards since both are essential to an effective conversion process.

5.1.2 What Is a Standard? A "Standard" is a consensus or common practice sometimes established by authority derived from a desire to reduce arbitrary variety for economic reasons. A standard is one method of insuring compatibility by using accepted conventions.

Several types of standards exist. For instance, a de facto standard is a practice accepted through common usage, although it has not been subjected to official standardization. On the other hand, standards have also been approved by an authorized official for use within a particular organization. For example, Federal Information Standards Processing (FIPS) are those that have been approved by the Federal standards process. American National Standards are those standards that have been approved for voluntary use by the American National Standards Institute (ANSI), a body made up of participants from government, industry, and academe. International standards are those that have been approved by the International Organization for Standardization (ISO) for voluntary use by member nations and international organizations.

In the case of FIPS, ANSI, and ISO, a formal mechanism has been established to propose, review, approve, and validate standards. While ANSI and ISO standards are voluntary for all participants, the FIPS standards are mandatory within the Federal Government. For mandatory standards to be effective, an enforcement and validation mechanism is necessary.

5.1.3 Background. As yet, no formal DBMS standards exist. Many groups concerned about standardization are actively working in this area. The committee felt that a review of the various Standards activities would be worthwhile for the reader. The Committee included in its review the Data Base Directions I Report and the activities being carried on by voluntary groups; specifically, the ANSI/X3/SPARC and CODASYL groups.

Data Base Directions I. Overall, the recommendations of the standards section of Data Base Directions I have two prominent characteristics: they are still valid, and they have not been implemented. Specifically, the Committee at that time recommended the undertaking of more international activity in data base standardization. Despite considerable interest in data base, standards are only now beginning to occur at the national or international level. Nothing comparable to the International Telegraph and Telephone Consultative Committee (CCITT) has been formed. Little progress has also gone on within the International Organization for Standardization (ISO). This lack of progress exists in spite of the various admonitions from

governmental operating bodies, such as Congressman Jack Brooks and the House Operations Committee in the United States, European Economic Community Committees, etc. In effect, no firm standards efforts have been undertaken. [Editors's note: subsequent to the preparation of this report but prior to publication. ANSI initiated several DBMS activities.]

Voluntary standards bodies such as ECMA, and American National Standards Institute/Standards Planning and Requirements Committe (ANSI/SPARC et al.) have been slow to agree upon firm recommendations for standards or interim action. The technical societies have not been able to recommend agreed upon and coherent roadmaps to the eventual realization of data base standards. The art and/or science of decision-making using data base data does not seem to have advanced. Data collection procedures may have improved since the days of manual actions with paper, but the science of decision-making based on such data does not seem to have made significant progress. For example: Data Base Directions I recommended the more accurate usage of terms. It suggested that "CDBS" (Computer Data Base System) be used instead of "DBMS" since organizations have always used data base. This suggestion underscores the important new factors to be considered by the reader of this new term. However, the suggestion has not had acceptance. On the positive side of the data base standards issue, substantial additions have been made to the literature, and we anticipate eventual, improved understanding of these issues.

Many practical data base conversions have been undertaken successfully -- both from conventional systems to a computer data base system, or from one computer data base system to another computer database system. A body of evidence indicating the economic superiority for the data base approach is accruing. The reader will read of such evidence in other sections of this report.

American National Standards Institute (ANSI). The Standards Planning and Requirements Committee (SPARC) of ANSI/X3 (Computers and Information Processing) established in 1972 a study group which was chartered to investigate the potential for standardization in the area of DBMS.

During early deliberations, the study group concluded that the proper subjects for DBMS standardization were interfaces within a DBMS. It was then necessary to define a framework for DBMS. In 1975, the study group produced an interim report [3] which described such a framework. A revised report has also been published recently [4] which represents the study group's most important ideas relevant to standardization.

The framework represents three kinds of objects: processing functions, person-roles, and interfaces between these. The objects were divided into three basic levels: the internal, the conceptual, and the external. The internal level is oriented towards the most efficient use of the computing facility. The conceptual level contains a logical representation of the persons, places, things, and their relations of interest to the organization using the DBMS. The external level contains different views of the data base which various applications need to do their information processing. The three levels are associated through explicit mappings. Mappings, through their ability to isolate changes to one level, facilitate data base conversion and form a basis for orderly standardization of DBMS. In addition, the ANSI/SPARC framework defines three administrator roles, one for each of the three levels in the framework. These distinct roles also are potential aids in standardization in that responsibilities are clearly delineated.

Although the study group itself did not recommend to its parent, ANSI/X3/SPARC, action for or against standardization of any DBMS component, SPARC recommended to ANSI/X3 that standardization efforts begin on a CODASYL Data Definition Language (DDL) and on CODASYL additions to FORTRAN and COBOL for data manipulation (DML). These efforts have been initiated.

CODASYL. Through the work of several technical committees coordinated by an executive committee, CODASYL has continued to develop specifications for a data definition language (DDL) and data manipulation languages (DML) for incorporation into the existing COBOL and FORTRAN languages. CODASYL has addressed a number of technical problems including concurrency and device independence and has endeavored, through task groups of its COBOL and DDL committees, to resolve inconsistencies between the language specifications produced by the two committees. CODASYL has authorized the publication of three journals of development (DDL JOD, COBOL JOD, FORTRAN JOD) containing the respective language specifications to facilitate the work of ANSI in defining standards for the three languages.

5.2 POTENTIAL BENEFITS THROUGH STANDARDIZATION

Standardization can provide many benefits to vendors and users involved in conversion tasks. In this sense, users mean the applications development people who will actually perform the conversion. The benefits can take many forms. Among the benefits the committee felt could be realized were:

- Improved Portability. Defining in a standard way the logical relationships of data would in particular insure that diverse hardware and/or software implementations could successfully address the same logical data. This could minimize hardware dependencies and allow the users to progress to both different and improved technology with a minimum expenditure of both time and money. Portability would provide protection against loss of the usually sizeable investment required by the user to establish a data base.

- Improved Education. Standards could provide a basis upon which educational institutions/corporations can structure their data processing training programs to satisfy the needs of their personnel. In effect, standards could reduce the time and expense of retraining data processing staffs.

- Enhanced Communication. Standard terms and definitions would enhance communication among users and/or vendors. As it now stands, these terms are being haphazardly developed and becoming widely used with little common understanding.

- Product Specification. Standards would provide product specifications that could be met by all vendors. End users will have, therefore, more precise product specifications and could exercise greater impact on what products would be manufactured to satisfy their needs.

- Easier to Specify User Requirements. The standardization machinery would enable the user community to be involved in the evolution of the data base technology. In this respect, it should be self-evident that the user would have a far greater interest than the vendor in ensuring that the standards applicable to data base reflect the need to minimize the problem of conversion.

5.3 SOFTWARE COMPONENTS IN CONVERSION PROCESS

This section examines the software components of the data base environment which the committee felt could be standardized. The rationale for standardization is based on four different conversion scenarios and the components which, if standardized, the committee felt would facilitate conversion. In all cases, technical feasibility of the standard has not been considered. The term "data base environment" is used to describe an environment in which the many parts making up an organization's data base would be available for shared use and coordination. This approach is less confusing than using the term "DBMS environment," since many installations are merely using the DBMS software as a substitute for traditional access methods. In this situation, the environment has not changed; it is merely a change of access method.

The term DBMS will be used to describe the software used to manipulate the computer resident portion of the data base. The conversion scenarios considered were:

- Moving from a non-data base environment to a data base environment using a standard DBMS.
- Moving from a standard DBMS to the same standard DBMS within a different hardware environment.
- Moving from a Type A standard DBMS to a different Type B standard DBMS.
- Moving from a standard DBMS to a non-standard DBMS environment.

5.3.1 Scenario 1. Moving from the non-data base environment to the standard data base environment. When considering the processes involved in data base conversion, several points can be identified which can be facilitated by the availability of standard tools or methods.

Data Flow Analysis. To determine the organization's data requirements, a data flow analysis should be undertaken. To facilitate this analysis a standard approach is needed. It should include a complete description of the data and where it is used, including all input and output documents, both manual and automated processes.

Dictionary/directory System (D/DS). Although it may be premature to consider a standard for a D/DS, this is an important tool for definition analysis and data flow analysis and to facilitate access to this information. A

more detailed discussion of the possible capabilities is presented elsewhere in the report.

Data Management Function. An organizational group concerned with overall responsibility for the management of data, both computerized and non-computerized. This function should have a specific functional description.

Data Base Administrator. A position holding primary responsibility for the overall accuracy, timeliness, and availability of the corporate data through direct control over the data dictionary/directory system. This position located within the data management function should have clearly understood responsibilities.

Loading the Computer Data Base. To facilitate the conversion effort, specific conventions are needed for placing data into data bases initially.

Conversion To the Data Base Environment This can be accomplished by either (1) moving current applications to the new environment with no major changes in process or (2) creating new applications. Both approaches use a Data Manipulation Language (DML) to access the data base from a user-oriented view of the database.

5.3.2 Scenerio 2. A DBMS to DBMS conversion where each DBMS is the same standard and only the hardware changes. In this case, the steps in scenario 1 were presumably implemented already. However, additional problems occur:

Data Translation. Moving the data in the source computer data base to the target computer data base. This requires a translation of the data from physical structure of the source computer to the physical structure of the target computer. A needed standard facility would be a method of unloading the physical data to a standard interchange format, in display format representation, and then loading that data into physical structure of the target computer.

Program Translation. Restatement of the application programs reflecting changes to the description of data are necessary as well as changes to the application programs. Standard data description and data manipulation functions are needed in addition to those stated in scenario 1.

5.3.3 Scenario 3. Conversion between standard versions of different DBMS. Assumes both standard DBMS incorporated the points in scenario 1. No additional points beyond those described above are needed, but significant differences in the details of the data models may occur.

5.3.4 Scenario 4. Standard DBMS to non-standard DBMS conversion. Similar to Scenario 3 because there would be no reason to make this step unless no standard DBMS satisfies information requirements.

5.3.5 Miscellaneous Standards Necessary. Additional computer components considered and mentioned as desirable, but for which standards may be premature or which may not affect the conversion effort are:

- Standard end-user facilities; i.e., a means whereby a non-computer professional can communicate with the data base.
- Network interfaces.
- Restart functions.
- Security functions.
- Communication facilities.
- Command Languages (operating systems).

5.3.6 Non-software Components Necessary. Many non-software components are involved in any conversion process. Those considered by the committee include:

Administrative/management Procedures. The committee identified no standards which could be applied to these procedures during conversion. However, the development of guidelines and checklists based on earlier experience and lessons learned would be very useful.

Verification of Compliance With Standards. The committee believed there is a need for a standard approach to validating compliance with DBMS standards. An important component of such a validation would be having a set of common procedures for measuring compliance.

5.4 RECOMMENDATIONS

Two areas should be considered immediately for standardization if progress toward successful conversions is to be made. These two areas are:

- Development of a standard DBMS.
- Development of a Generalized Dictionary/Directory System.

5.4.1 The Development of a Standard DBMS. A DBMS standard would benefit government and industry in their conversion efforts and the Committee encourages progress toward a DBMS standard. The work of CODASYL group, with some possible modifications, appears to offer a first step toward the complete specification of a DBMS, although some committee members felt several sucessful commercial DBMS may become de facto standards. The committee also believed that consideration should be given to the idea that the standard should not inhibit any future technological developments in hardware and software.

5.4.2 Generalized Dictionary/directory System. The use of a Dictionary/Directory facility is highly desirable as a tool in any conversion process, whether from manual or conventional file to a data base environment or from one DBMS to another DBMS. The advantages for the use of a dictionary facility will be gained primarily in the following two areas:

- It will allow the proper assessment and definition of the organization's data to be undertaken or to be made prior to the start of a project.
- It provides an essential management mechanism to control the actual development of the new system.

Since, in this sense, the dictionary facility chronologically precedes the DBMS targeted in the conversion, it appears likely that the direct support of the dictionary facility itself should not depend on the target DBMS, or, for that matter, on any particular DBMS. In any case, the dictionary facility must be able to support descriptive facilities for the data in a logical mode.

Another useful capability is that of being able to produce processable data descriptions in the target DBMS environment. This does not, however, imply that this target DBMS must be used in the implementation of the dictionary facility, but rather that a suitable (possibly manual)

interface to this DBMS be made available. It would be highly desirable, if not required, that all DBMS had the same logical interface. Then the dictionary could interface with multiple DBMS at the same time.

In discussing the architectural placement of the dictionary, this facility should not exist as an adjunct to any DBMS, but as a separate facility that interfaces to data in conventional files as well as to data being managed by one or more DMBS. A distinction needs to be made in the manner in which the dictionary facility is invoked;

- In the case of a free-standing dictionary, this facility is invoked under user control. As such, it is a user option if, and when, the dictionary is to be invoked in the execution of any process.

- In the case of an integrated dictionary, which means the data dictionary is directly available to the DBMS, this facility might be invoked automatically by the system itself in the execution of any process, thus assuring synchronization of the process with the information contained in the dictionary. The feasibility of this approach depends on the overhead burden it may generate.

Since the use of some of the currently available dictionary facilities is believed to be of considerable value in the attainment of these and other goals, it is visualized that a more general facility, called a Generalized Dictionary/Directory System, would be more effective. In brief, a GD/DS is an information system in and of itself whose subject matter is all the information about the enterprise on the following classes of entities:

- Data Components.

- Processing Components.

- People and Organizational Components.

- Events.

Attributes to be included in the data dictionary are those about the entities themselves, the relationship that exists among the entities, as well as the context in which these relationships exist. For example, among the attributes would be the relationship between systems and both automated and manual procedures. The generalized data dictionary/directory should contain not only the logical attributes and relationships between the entitites described, but also attributes associated with the physical

location of the entities.

5.5 CONCLUSION

In summary, the committee strongly believes that many areas for standardization exist within the conversion process and the data base environment in general. The committee notes with frustration the lack of overall progress since the first Data Base Directions workshop.

The committee noted that in actual use DBMS are often misapplied. It is possible, however, that this misuse relates to the historical problems in the environments when DBMS are introduced. For example, most organizations seem to be obtaining DBMS as their initial introduction into the data base environment and then find a need for a Dictionary/Directory which is then introduced as a secondary action. Only when both of these steps do not really solve the problem, does the organization look at its total organization and address the fundamental work necessary. This fundamental work includes documenting data, systems, processes, etc., to insure their more effective operation in this new DBMS environment.

The committee observed that the installation process actually should be reversed with the data dictionary introduced first. All of the data within the environment and the systems that use them should be carefully documented and defined (including all the developmental shortcuts that had been taken in the past), and then entered into a Generalized Dictionary/Directory. Once this is accomplished, then the appropriate data bases to support their various information systems in the organization can be built.

5.6 REFERENCES

The following references are included as part of the standard panel Report as a convenience to the readers. Some of the material included was not used directly by the committee but did influence the thinking of some of the participants.

1. Sibley, Edgar H., "Standardization and Databaase Systems," IFSM TR No. 23, University of Maryland, College Park, MD 20742, July 1977

2. Berg, John L., (Editor), "Data Base Directions - the Next Step," Proceedings of the Workshop of NBS and ACM held at Ft. Lauderdale, FL, October 29-31, 1975, National Bureau of Standards Special Publication 451

3. ANSI/X3/SPARC Study on Data Base Management Systems, "Interim Report, 75-02-08," published as Vol. 7, No. 2 1975 of FDT, the Bulletin of ACM-SIGMOD

4. Tsichritzis, D. and A. Klug (eds.) "The ANSI/X3/SPARC DBMS Framework: Report of the Study Group on Data Base Management Systems," AFIPS Press, 1977.

5. CODASYL Data Description Language Committee, "Data Description Language," Journal of Development, Jan. 1978.

6. Leong-Hong, B. & Marron, B., "A Technical Profile of Seven Data Element Dictionary/Directory Systems," NBS Special Publication 500-3, February 1977

7. Lefkovits, Henry C., "Data Dictionary Systems," Q.E.D. Information Sciences, 1977

8. CODASYL Data Description Language Committee, "DDL Journal of Development," June 1973, NBS Handbook 113, January 1974, Page 155

9. "Data Dictionary/Directory Evaluation Criteria," M. Bryce & Associates, Inc., Cincinnati, Ohio

10. "Installing a Data Dictionary," EDP ANALYZER, Vol. 16, No. 1, January 1978

6. CONVERSION TECHNOLOGY--AN ASSESSMENT

James P. Fry

CHAIRMAN

Biographical Sketch

James P. Fry is the Director of Database Systems Research Group in the Graduate School of Business Administration at The University of Michigan where he directs research in areas of database design, distributed systems, database restructuring and systems conversion. Previously he held positions at the MITRE Corporation, The Boeing Company, and E. I. du Pont de Nemours.

Nationally listed in Who's Who in Computer Research and Education and having published over thirty-five articles, Mr. Fry is a well known speaker at AFIPS and IFIPS conferences and an invited lecturer to others. Professionally he has participated in the CODASYL organization (on the Systems Committee and Database Program Disk Group), the Association for Computing Machinery, National Chairman Special Interest Group on Management of Data and the SHARE/GUIDE User Organization.

Participants

Edward Birss
Peter Dressen
Nancy Goguen
Michael Kaplan
Eugene Lowenthal
Vincent Lum
Robert Marion
Shamkant Navathe
Steven Schindler
Arie Shoshani
Stanley Su
Donald Swartwout
Robert Taylor
Beatrice Yormark

6.1 INTRODUCTION

6.1.1 The Scope of the Conversion Problem. Two factors make conversion necessary: changes in users' functional requirements and changes in their performance requirements. These changes may necessitate the acquisition of new software and hardware which, in turn, may require changes to the existing data programs. For example, the acquisition of a data base management system to replace a file management system requires the integration of the original files into a data base system and modification of the programs to interact with it. The replacement of a current DBMS with a new data base system to provide additional functionality may require a different way of logically structuring the information (its data model) and a different kind of language interface. The establishment of a communication network between differing systems to implement data sharing will require dynamic (i.e., in real time) conversion of data between different nodes in the sense that the same item will repeatedly undergo conversion as it is needed, or alternatively, it requires conversion of programs when they travel to different nodes to access data there. Even when the acquisition of new software or hardware is not warranted, changes in a users' functional and performance requirements can require data and program conversion.

What makes conversion difficult is the proliferation of data models and levels and styles of DBMS interfaces, internal data representation, and hardware architecture. This panel will examine the technology that has been developed to perform conversions, analyze the areas which require new or improved techniques, and consider strategies for minimizing the need to convert as well as for streamlining the unavoidable conversions. Over the past six years, research and development has primarily centered on the problem of converting in non-dynamic environments. The first part of the section called "CONVERSION TECHNOLOGY" surveys tools and technology currently available for the conversion of data organization. More recently, data base program conversion--probably the most difficult part of the conversion problem--has received attention. The middle part of this section considers the directions being taken by current data base conversion research. The final part analyzes the status of the entire technology. The third and final section explores the factors affecting conversion, the approaches for reducing the need to convert data and applications programs, and the impact of new software and hardware technologies on conversion. Section 4 summarizes the trends in conversion needs and tools.

6.1.2 Components of the Conversion Process. When conversion is necessary, users will extract data from their source environment and restructure them to the form required for the target environment. While the extraction and restructuring may themselves be complex, these processes are frequently complicated further by the undisciplined nature of the source data. For example, data may exist in duplicate or contain numerous errors and multiple inconsistencies. The whole process of extracting it from its source, "cleaning the data," restructuring it to a desired form, and loading it into the targeted environment is generally referred to as data conversion or translation.

After a data conversion, particularly one involving extensive restructuring, the application programs which process the original data may not run correctly against the new data. A small amount of restructuring may require only a simple modification, while extensive restructuring may require extensive rewrite or redesign. The process of modifying the programs to process the restructured data is referred to as application program conversion or translation. (This report will not consider the problem of program conversions not related to a change in data structure.) Let us discuss data conversion and application program conversion a bit further.

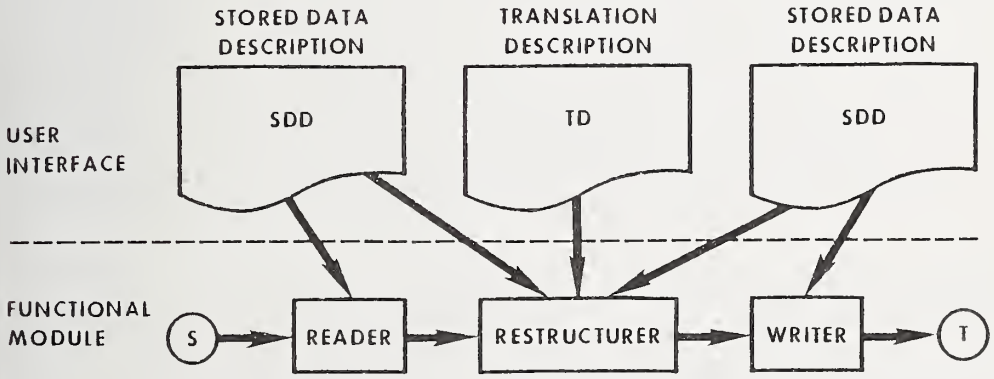


Figure 6-1
A Data Conversion Model

Concepts of Data Conversion

In brief and conceptual terms, the data translation or conversion process can be represented diagrammatically in Figure 6-1. As shown in this diagram, a data translation system generally requires three components: a reader, a restructurer, and a writer. While the capability of each component depends on the individual design of a data translation system, the reader, in general, accesses data from its source environment to prepare it for further processing. The accomplishment of this process unquestionably requires a description of the data and, thus, a data description language. The writer is the functional inverse of the reader, and puts the transformed data into the target environment. It too requires a description of the data structure and shares with the reader the need of a data description language. The restructurer functions quite differently from that of the other two components. This component, in general, extracts data from its source or internal forms and restructures it to a desired format or structure. This process usually requires a translation description language.

In a data conversion system with a limited application intended, the three components may not be distinguishable, nor the need for the two languages clear. For example, if one wishes to create a conversion system merely to translate EBCDIC characters into ASCII, one can create a simple system with one component and a simple data description language embedded in it. However, if development of a broadly applicable data translation system is the goal, clearly one must have a reader and writer capable of accessing and putting out data in all kinds of environments and a powerful restructurer capable of all manners of manipulating data and of creating some data as well. Such a generalized system implies the need for versatile data translation description languages.

Concepts of Data Base Program Conversion

Figure 6-2 represents a general approach to data base program conversion or translation. To convert a program one must determine the functions of the program, and its semantics. Programmers making assumptions about the state of the data may not, and currently need not, state these assumptions explicitly in their programs. Therefore, one usually must provide more information about the semantics of the program than that provided in the program text and its documentation. The program conversion process also needs information about the data structure the program originally ran against, the new structure it must run against, and how the two relate. These data descriptions could be the same

as those used to drive the data translation process. The program, the description of its semantics, and the description of the data conversion are inputs to the program conversion process. It uses them to determine the data originally accessed and how to accomplish the same access in the new structure, and it produces a new program to do this. Currently, a combination of manual translation, emulation, and bridge programs accomplishes this conversion process and an automatic or semi-automatic program conversion technology is only in the early stages of research.

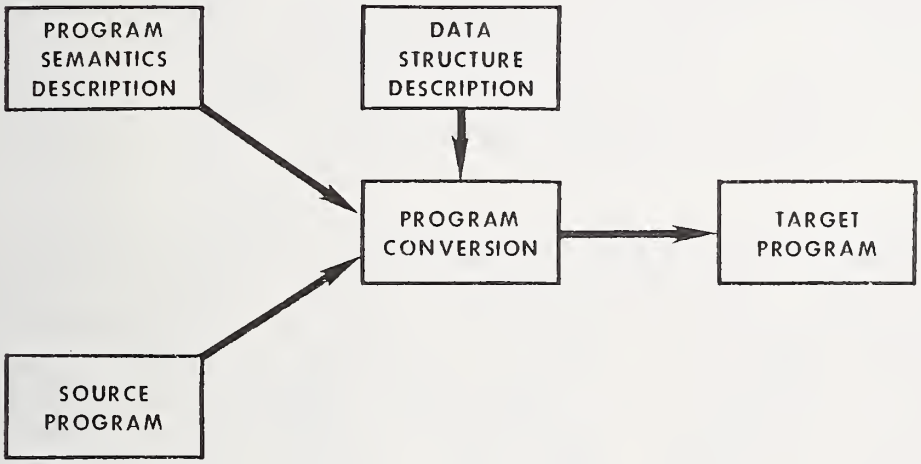


Figure 6-2
Program Translation

6.2 CONVERSION TECHNOLOGY

This section turns from the general concepts of data and program conversion to the technology by which a conversion is achieved. Since most of the conversion results have been achieved in data conversion, we focus on this aspect and limit our discussion on program conversion to those cases which are the result of a data conversion.

6.2.1 Data Conversion Technology. Currently, as the common approach to data conversion, one develops customized programs for each transfer of data from one environment to another. This approach is inherently expensive: the programs are developed for use only once and their development costs cannot be amortized. Further, poor reliability results from the greater likelihood of making errors as data is passed from program to program. For a large data conversion effort, tracing data back through several passes to the source of an error in the data at a particular point can become unmanageable. As an alternative one may search for a broader approach to data conversion with a generalized system. We shall now describe such systems in more detail.

Problem Discussion. Data exists in many varied and complex forms; Figure 6-3 (next page), an expanded form of the diagram in Figure 6-1, indicates some of the transformations that need to take place in a data conversion. This diagram illustrates the capabilities needed by the read and write process in a data conversion system.

Unloading of data from its source permits reducing the complex physical structure of the source data base to a very simple physical structure; namely, a sequential data stream. The source data base contains not only the information that interests the user, but also a large amount of control information specific to a particular system. This control information is used by the system for such data management functions as overflow chaining, index maintenance, and record blocking. For example, many systems mark a record to be deleted rather than actually delete it and the unload transformation can remove such system specific information. Another factor that causes complexity in the unloading process is the frequent use of pointers in a source system. Pointers are used for two basic purposes: (1) to represent relationships that exist between record instances, and (2) to implement alternative access paths to the data. During the unload transformation, the second class of pointers may be discarded without loss of information. The first class of pointers, however, contains information that the transformation must preserve.

The reformat process creates a common data form of the source data for the restructuring step in the conversion process. In the case of a simple sequential file, not under DBMS control, it may enter the conversion process at this point. Depending on the design of a system, this step may involve editing (i.e., encoding or recoding of items), limited data extraction, correction, and the like.

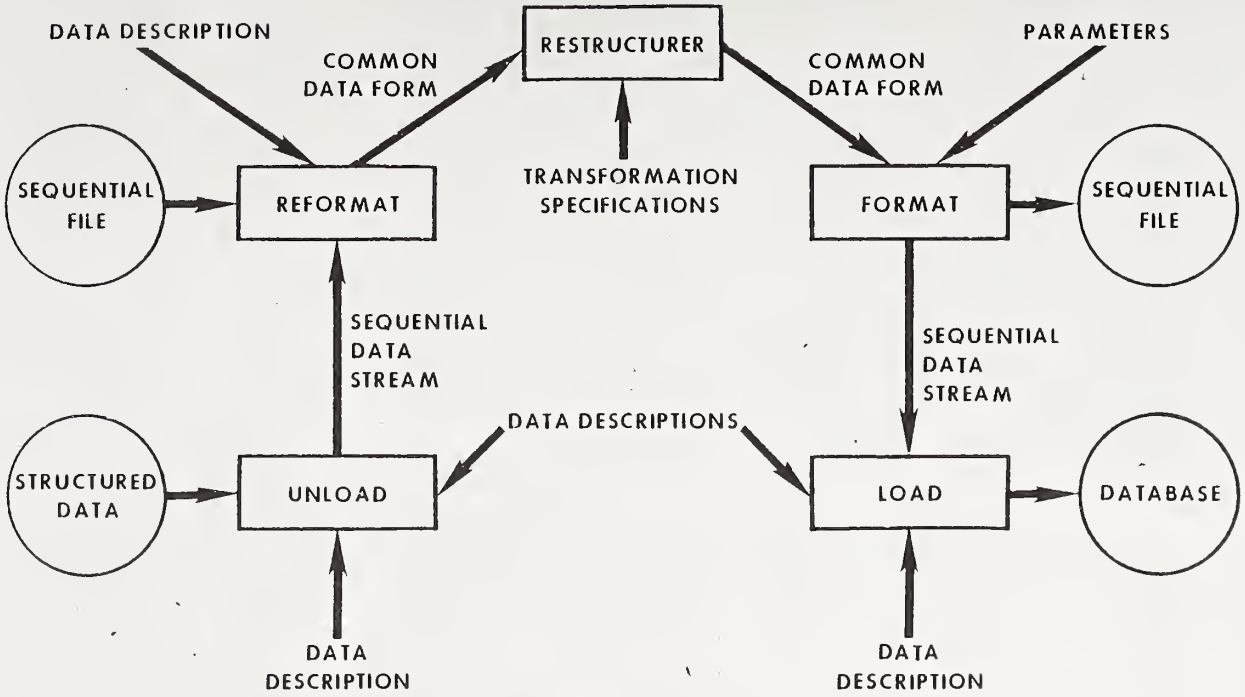


Figure 6-3
Schematic Data Conversion Process

Since this step seeks to create a data structure of the source data without system-dependent information, one can consider the mapping between the input and the output of the reformat process to be generally one-to-one. While this step looks simple functionally, its actual application and implementation can be quite complex. For example, an application program may use the high order bits of a zoned decimal number for its own purposes, knowing that these bits are not used by the system. Such specifications of nonstandard item encodings present a difficult problem in data conversion.

The load process is the counterpart of the unload process and needs no further clarification. Note, however, that the use of a common data form provides additional benefits, such as easing the portability problem.

The restructuring process undoubtedly represents the most complex process of a generalized data conversion system. The languages for this mapping process can differ widely (for example, some procedural and other nonprocedural) and the models used to represent the data in the conversion system are also quite divergent. (For example, some use network structures; others use hierarchical structures). More will be said on this topic later in this section.

Let us now turn to discuss the issue of implementation briefly. Generally, there are two techniques: an interpretive approach or the generative approach. In the interpretive approach, the action of the system will be driven by the descriptions written in the system's languages via the general interpreter implemented for the particular system. In the generative approach, the data and mapping descriptions are fed into the compiler(s) which generates a set of customized programs executable on a certain machine. Later in this section we'll discuss the merits of each of these approaches.

Turning our attention to the tools that have been developed for data conversion, we shall first discuss currently available tools and then the research and development work in progress.

Available Conversion Tools. Currently, available tools have limited capabilities. Because it is impossible in this short report to provide an exhaustive survey of all the vendor-developed conversion tools, we will highlight the spectrum of capabilities available to the user by providing examples from specific vendor shops.

The repertoire of vendor conversion tools begins at the character encoding level of data conversion with the provision of hardware/firmware options and continues through the software aids for conversion and restructuring of data bases.

Depending on a diversity of conditions, the need to develop software tools varies from vendor to vendor. Probably the most prevalent file conversion tool is a COBOL foreign file processing aid. This type of facility allows the direct reading or writing of a particular class of files such as EBCDIC tapes or Honeywell Series 200/2000 files within COBOL. Although a relatively widespread facility, its capabilities are nevertheless limited. For example, some do not handle unlabeled tapes, while others cannot process mixed mode data types. Aside from the work of Behymer and Bakkom [DT11], which was aimed toward achieving a general conversion bridge with a particular vendor, to our knowledge there are no vendor supported generalized file translation tools.

In contrast to the above file translation tools, tools have been developed that have their main applications in a data base environment. One example of a data base conversion aid is the I-D-S/II migration aid provided by Honeywell. Because of the large volumes of data involved and the fact that the user cannot afford to shut down his whole data processing shop, a co-existence approach was adopted. The first step is to reformat the I-D-S/I data base into the I-D-S/II format, making the necessary data type conversions and pointer mechanism adaptions. This step allows the data base to be processed in the I-D-S/II mode, but not optimally. Additional steps in this migration include the generation of the additional I-D-S/II pointer fields (I-D-S/II requires "Prior & Header" chain pointers, which are allocated in step 1 but not filled in) and the restructuring of the I-D-S/II (Coexistence) to the more sophisticated capabilities of I-D-S/II.

Some data base restructuring tools specific to a particular DBMS have been developed by DBMS users. One example of this type of tool is REORG [R11], a system developed at Bell Laboratories for reorganization of UNIVAC DMS-1100 data bases. REORG provides capabilities for logical and physical reorganization of a data base using a set of commands independent of DMS-1100 data management commands. A similar capability has been developed at the AllState Insurance Company.

In addition to the above, there are also software companies and vendors who will do a customized conversion task on a contractual basis.

Data Conversion Prototypes and Models. Over the past seven years, a great deal of research on the conversion problem has been performed, with the results summarized in Figure 6-4. The University of Michigan, the University of Pennsylvania, IBM, SDC, and Bell Laboratories initiated projects, as well as a task group of the CODASYL Systems Committee. In many cases, interaction and cross-fertilization between these groups led to some consensus on appropriate architectures for data conversion. The individual achievements of these groups is discussed below:

The CODASYL Stored-Data Description and Translation Task Group In 1970, the CODASYL Systems Committee formed a task group (originally called the Stored Structure Description Language Task Group) to study the problem of data translation. The group presented its initial investigation of the area in the 1970 SIGMOD (then SIGFIDET) annual Workshop in Houston [SL1]. In 1972, the group was reformulated as the Stored-Data Description and Translation Task Group and presented a general approach to the development of a detailed model for describing data at all levels of implementation [SL4,DT2]. The most recent work of the group specifies the data conversion model and presents an example language for describing and translating a wide class of logical and physical structures [SL8]. The stored-data definition language allows data to be described at and distributed to the access path, encoding, and device levels.

The University of Michigan. The nonprocedural approach to stored-data definition set forth by Taylor and Sibley [SL3,6] provided one of the major foundations for the development at the University of Michigan (see Figure 6-4) of data translators. In concert with Taylor's language, Fry, et al. [DT1] initiated a model and design for a generalized translation.

The translation model was tested in a prototype implementation of the Michigan Data Translator in 1972 [UT2,4], and the results of the next implementation, Version I, were reported by Merten and Fry [DT4].

In 1974, the work of the Data Translation Project of the University of Michigan focused on the data base restructuring problem. Navathe and Fry investigated the hierarchical restructuring problem by developing several levels of abstractions, ranging from basic restructuring types to low level operations [R6].

	CODASYL STORAGE STRUCTURE DEFINITION LANGUAGE TASK GROUP:	UNIVERSITY OF MICHIGAN:	UNIVERSITY OF PENNSYLVANIA:	IBM RESEARCH, SAN JOSE:	SYSTEM DEVELOPMENT CORPORATION:
1970	Specifications of the data conversion problem and possible approaches to solving it [SL]				
1971		Model of logical structure structure to physical structure mappings for DBMS's [SL3,6]	Model and languages for Stored-Data description and translation description [SL2] (Moore School)		
	Stored-Data Description and Translation Task Group:	Data Translation Model [DT1]			
1972	Model of generalized data conversion architecture [SL4,DT2]				
1973		Restricted prototype data translator (sequential files) [UT3,4,DT4]	Restricted prototype data translator [DT3,6] (Moore School)	Model and Language for data translation (AIM[DT5] Define [SL7])	
1974		Model of restructuring [R6]			Data translation prototype [R3,4]

Figure 6-4
Historic Context of Data Conversion Efforts

Figure 6-4 (continued)
Historic Context of Data Conversion Efforts

	CODASYL STORAGE STRUCTURE DEFINITION LANGUAGE TASK GROUP:	UNIVERSITY OF MICHIGAN:	UNIVERSITY OF PENNSYLVANIA:	IBM RESEARCH, SAN JOSE	BELL LABORATORIES:
				Convert [R2]	
1975			External test of Penn Translator [TA1] (Moore School)		
			Scope of project broadened to generation of business systems		
1976	Detailed model of stored data [SL8]	Michigan Data Translator Operational Prototype (network restructuring and implementation of Translation Description Language [DT13,16]	Dynamic restructuring [R8] (Wharton School)	Implementation of XPRS [DT15]	
	Group redirected attention to program conversion				
1977		Access Path Specification Language [R2]		Laboratory of external tests of XPRS	ADEPT Current development data translation [DT17]

Later, Navathe proposed a methodology to accomplish these operations using a relational normal form for the internal representation [DT12]. Version II of the Michigan Data Translator was designed to perform hierarchical restructuring transformations, but the project did not implement it. Instead, the research was directed into the complex problem of restructuring network type data bases. To address this problem, Deppe developed a dynamic data model--the Relational Interface model--which simultaneously allowed a relational and network view of the data base [UR3]. This model formed the basis of the Version IIA design and implementation of generalized restructuring capabilities [UT8,9,10]. Another component necessary for the development of a restructurer was the formulation of a language in which to express the source to target data transformations. This language, termed Translation Definition Language (TDL), evolved through each translator version beginning with a source-to-target data item "equate list" in the Version I Translator to the network restructuring specifications of Version IIA. While the initial version of the TDL was quite simplistic, the current version, the Access Path Specification Language [DT16,R9], provides powerful capabilities for transforming network data bases.

The University of Pennsylvania. Concurrent with the work at the University of Michigan, Smith at the University of Pennsylvania (see Figure 6-4) also took a data description approach and developed a stored-data definition language (SDDL) for defining storage of data on secondary storage devices, and a translation description language (TDL) [SL2,DT2] and three levels of data base structures, the logical, storage, and physical, are described using the SDDL. In order to describe the source-to-target data mappings, a first order calculus language was used. Following from this work, Ramirez [DT3,6] implemented a language-driven "generative" translator which created PL/1 programs to perform the conversion. One of the first reports on the utilization of generalized translation tools was provided by Winters and Dickey [TAI]. Using the translator developed by Ramirez, they installed it on their system, and applied it to converting IBM 7080 files.

IBM Research, San Jose In 1973, another major data translation research endeavor was initiated at the IBM Research Laboratory in San Jose, California. Researchers in this project--initially Housel, Lum, and Shu, later joined by Ghosh and Taylor--adopted the general model as specified in Figure 6-1 but made several innovations. First, in the belief that programmers know well the structure of the data in a buffer being passed from a DBMS to the application program, the group concentrated its effort on designing a

data description language appropriate for describing data at this stage. Second, regardless of the data model underlying any DBMS, the data structure at the time it appears in the buffer of an application program will be hierarchical. The general architecture, methodology, and languages reflecting these beliefs is reported in Lum et al.[DT14].

In addition, the group in San Jose felt that, while it is desirable to have a file with homogeneous record types, it is a fact of life that many of today's data are still in COBOL files in which multiple record types frequently exist within the same file. As a result, the group concentrated on designing a data description language which can describe not only hierarchical records (in which a relational structure is a special case) but also most of the commonly used sequential file structures. This language, DEFINE, is described by Housel et al.[SL7].

The philosophy of restructuring hierarchies is further reflected in the development of the translation definition language CONVERT, as reported by Shu et al [R2]. This language, algebraic in structure, consists of a dozen operators, each of which restructures one or more hierarchical files into another file. The language possesses the capability of selecting records and record components, combining data from different files, built-in functions (e.g., SUM and COUNT), and the ability to create fields and vary selection of the basis of a record's content (a CASE statement).

A symmetric process occurs at the output end of the translation system. Sequential files are created to match the need of the target loading facility. The specification of this structure is again made in DEFINE.

A prototype implementation, originally called EXPRESS but renamed XPRS, is reported in [DT15].

System Development Corporation Another restructuring project reported by Shoshani [R3,4] was performed at The System Development Corporation in 1974-1975. In order to avoid the complexities of storage structure specification (i.e., indexes, pointer chains, inverted tables, and the like) they chose to use existing facilities of the systems involved. In particular, they advocated the use of query and load (generate) facilities of data base management systems. However, when such facilities do not exist, reformatters from the source (e.g., index sequential file) to a standard form and from the standard form to same output file had to be used. Given that data bases can be reformatted to and from a standard form, they concentrated on the problem of logical restructuring of hierarchical data

bases in this form.

The language used in the above project for specifying the restructuring functions (called CDTL--Common Data Translation Language) was designed to be conceptually simple. For the most part, it provides functions for specifying a mapping from a single field (or combination of fields) of the source to a single field of the target. For example, while a DIRECT would specify a one-to-one mapping of source items to target items, a REPEAT would specify the repetition of a source item for all instances of a lower level in the target hierarchy. In both cases, only the source and target fields need to be mentioned as parameters. In addition, there are more global operations, such as the INVERSION operator, which causes parent/dependent record relationships to be reversed. The system also supported extensive field restructuring operators, where individual field values could be manipulated according to prescribed language specifications. Since most of these operators are local, there is the possibility that they could be used in combinations that do not make sense globally. Therefore, a further component of the system was built to perform "semantic analysis," which checks for possible inconsistencies before proceeding to generate the target data base.

Bell Laboratories The Bell Labs data translation system ADAPT (A DAta Parsing and Transformation system), currently under development, is a generalized translation system driven by two high-level languages [DT17]. With the first language one describes the physical and logical format and structure of the data and to provide various tests and computations while parsing the source data and generating the target data. The second language is used to describe the transformations which are to be applied to the source data to produce the target data. Extensive validation criteria can be specified to apply to the source and target data.

Two processing paths are available within the ADAPT system: a file translation path and a data base translation path (see Figure 6-3). A separate path for file translation responds to real-world considerations: many types of conversions do not require the capabilities and associated high overhead involved in using a data base translation path.

Related Work Additional research effort examines the development and acceptance of a standard interchange form. An interchange form would increase the sharing of data bases and provide a basis for development of generalized data translators. The Energy Research and Development

Administration (ERDA) has been supporting the Interlaboratory Working Group for Data Exchange (IWGDE) in an effort to develop a proposed data interchange form. The proposed interchange form [GG2] has been used by several ERDA laboratories for transporting data between the laboratories. Additional work on development of interchange forms has been pursued by the Data Base Systems Research Group at the University of Michigan [UT14].

Navathe [R10] has recently reported a technique for analyzing the logical and physical structure of data bases with a view to facilitating the restructuring specification. Data relationships are divided into identifying and nonidentifying types in order to draw an explicit schema diagram. The physical implementation of the relationships in the schema diagram is represented by means of a schema realization diagram. These diagrammatic representations of the source and target data bases could prove to be very useful to a restructuring user.

6.2.2 Application Program Conversion. So far, we have concentrated on the data aspects of the conversion problem; it is necessary to deal as well with the problems of converting the application programs which operate on the data bases. Program conversion, in general, may be motivated by many different circumstances, such as hardware migration, new processing requirements, or a decision to adopt a new programming language. Considerable effort has been devoted to special tools such as those to assist migration among different vendor's COBOL compilers, and general purpose "decompilers" that have been developed to translate assembly language programs to equivalent software in a high level language. While progress has been made developing special purpose tools for a limited program conversion situation, little progress has been made in obtaining a solution to the general problem of program conversion. With this fact in mind, this section focuses on the modifications to application programs that arise as a consequence of data restructuring/conversion.

Problem Statement. There are three types of data bases which can affect application programs:

- alterations to the data base physical structure, for example, the format and encoding of data, or the arrangement of items within records
- changes to the data base logical structure--either:

  a. the deletion or addition of access paths to accommodate new performance requirements, or

b. changes to the semantics of data, for example, modification of defined relationships between record types or the addition or deletion of items within records

- migration to a new DBMS, perhaps encompassing a data model and/or data manipulation language different from the one currently in use

The actual impact of these data base changes on application programs is a function of the amount of data independence provided by the Data Base Management Systems. Data independence and its relationship to the conversion problem are discussed elsewhere [GG3]. We assume here incomplete data independence and that therefore some degree of program conversion is required in response to data base schema changes. In fact, whereas most commercial data base management systems provide application programs with insulation from a variety of modifications to the physical data base, protection from logical changes--particularly at the semantic level--is minimal. Examples of semantic changes likely to have a profound effect on application programs include:

- Changes in relationships between record types, such as changing a one-to-many association to a many-to-many association or vice-versa.

- Deletion or addition of data items, record types, or record relationships.

- Changing derivable information ("virtual items") to explicit information ("actual items") or vice-versa.

- Changes in integrity, authorization or deletion rules.

Various properties of data base application programs greatly complicate the conversion problem. For instance, many data base management systems do not require that the record types of interest (or possibly even the data base of interest) be declared at compile time in the program; rather these names can be supplied at run time. Consequently at the compile time, incomplete information exists about what data the program acts on. Other troublesome problems occur when programs implicitly use characteristics of the data which have not been explicitly declared (e.g., a COBOL program executes a paragraph exactly ten times because the programmer knows that a certain repeating group only occurs ten times in each record instance). Complexity is introduced whenever a data manipulation language is

intricately embedded in a host language such as COBOL. The interdependence between the semantics of the data base accesses and the surrounding software greatly complicates the program analysis stage of conversion. Because of these considerations, substantial research has been devoted to alternatives to the literal translation of programs. In particular, some currently operational tools utilize source program emulation or source data emulation at run time to handle the problem of incomplete specification of semantics and yet still yield the effects of program conversion.

Current Approaches. In this section, we discuss two main techniques currently employed in the industry. These techniques are commonly used but unfortunately not documented in the form of publications.

DML Statement Substitution

The DML substitution conversion technique, which can be considered an emulation approach, preserves the semantics of the original code by intercepting individual DML statement calls at execution time, and substituting new DML statement calls which are correct for the new logical structure of the data base. Two IBM software examples which provide this type of conversion methodology are 1) the ISAM compatibility interface within VSAM (this allows programs using ISAM calls to operate on VSAM data base), and 2) the BOMP/DBOMP emulation interface to IMS. This program conversion approach becomes extremely complicated when the program operates on a complex data base structure. Such a situation may require the conversion software to evaluate each DML operation against the source structure to determine status values (e.g., currency) in order to perform the equivalent DML operation on the new data base. Generalization of this approach requires the development of emulation code for the following cases: maintain the run time descriptions and tables for both the original and new data base organizations, intercept all original DML calls, and utilize old-new data base access path mapping description (human input) and rules to determine dynamically what set of DML operations on the new data base are equivalent to each specific operation on the source data base.

Although a conceptually straightforward approach, it has several drawbacks. The drawbacks can be categorized as degraded efficiency and restrictiveness. Efficiency is degraded primarily because each source DML statement must be mapped into a target emulation program, which uses the new DBMS to achieve the same results. The increased overhead in program size and/or processing requirements can be significant.

The drawback of restrictiveness comes about because the emulation approach inhibits the utilization of the increased capabilities of the new DBMS and/or data structure through the modeling of the old methods. Additionally dependence upon the old program semantics limits the sets of permissible new data structures that must support all of the semantics of the source program if the source program is to continue to execute in the same manner. Note that the rules can be quite complex, even for the limited situation of which the data structure changes preserve semantic equivalence. Therefore, in some instances, just the limited task of determining if a change in data structure (given no change in the data model) will support a set of source programs will be an extensive task.

Bridge Program The second method in use today is sometimes referred to as the Bridge Program Method. In this technique, the source application program's access requirements are supported by reconstructing from the target data base that portion of the source data base needed. Data reconstruction is done by means of "bridge programs." The source program is then allowed to operate upon this reconstructed portion of the source data base to effect the same results that would occur if the source data base were not modified. Of course, a reverse mapping is required to reflect and update, and each simulated source data base segment must be prepared before it is needed by the application program.

This approach suffers from the same types of disadvantages inherent in the emulation approach. Efficiency problems for complex/extensive data bases and programs performing extensive data accessing can make this method prohibitively expensive for practical utilization. This technique is generally found as a "specific software package" developed at a computer installation rather than as a standard vendor supplied package.

Current Research. Differing from the emulation and bridge program approaches, current research aims towards developing more generalized tools to automatically or semi-automatically modify or rewrite application programs. The drawbacks of the existing approaches described above can be avoided by rewriting the application programs which would take advantage of the new structure and semantics of a converted data base and by using a general system to do the conversion rather than using ad hoc emulation packages and bridge programs.

Research on application program conversion is still in its infancy. Consequently, very few published papers on this subject exist. This section describes a handful of works in the order of the dates of publication. Mehl and Wang [PT6] presented a method to intercept and interpret DL/1 statements to account for some order transformations of hierarchical structures in the context of the IMS system. Algorithms involving command substitution rules for various structural changes have been derived to allow the correct execution of the old application programs. This approach works only for a limited number of order transformation of segments in a logical IMS data base. Since it is basically an emulation approach, it has the drawbacks discussed in the previous section.

A paper by Su [PT12] gives a general model of application program conversion as related to data base changes resulting from a data base transformation. An attempt was made to identify the tasks required for the automatic or semi-automatic conversion of application programs due to data base changes. The paper stresses two main points: 1) the need for extensive analysis of an application program including the analysis of program logic, data variable relations, program-subprogram structure, execution profile, etc.; and 2) the use of data base translation operators to determine what program transformations are required to account for the effects of these operators. The idea of the use of a common language to describe the operations of source queries and the data translation statements is also proposed.

An approach to the transformation of DBTG-like programs in response to a data base restructuring was proposed by Schindler [PT10]. The approach is based on the concept of code templates, which are predefined sequences of host language--DML statements (roughly analogous to assembly language macros). Application programs can be written as nested code templates. The code templates are devised so that each one corresponds to an operator in the relational algebra. An application program is then mapped into a relational expression, transformations are performed on the expression to accommodate the data base restructuring, and a new program is generated by mapping the transformed expression back into code templates. This approach suggests that a level of logical data independence may be achieved through current programming technology.

The work by Su and Reynolds [PT15] studied the problem of high-level sublanguage query conversion using the relational model with SEQUEL [Z5] as the sublanguage, DEFINE [SL7] as the data description language and CONVERT [R2] as the translation language. Algorithms for rewriting the

source query were derived and hand simulated. In this study, query transformation is dictated by the data translation operators which have been applied to the source data base. The purpose of this work was to study the effects of the CONVERT operators on high-level queries. Only restricted types of SEQUEL queries were considered. This work demonstrates that a general program conversion system should separate the data model and schema dependent factors from the data model and schema independent factors; and an abstract representation of program semantics and the semantics of data translation operators need to be sought so that data conversions at the logic level (especially the type which changes the data base semantics) and the DBMS level can be attempted.

Two indpendent works carried out about the same time by Su and Liu [PT13] and Housel [PT14] take a more general approach to the application program conversion problem. The former work is based on the idea that the same data semantics (a conceptual model) can be modelled externally by various existing data models (relational, hierarchical and network) using different schemas. Application programs are mapped into an abstract representation which represents program semantics in terms of the primitive operations (called access patterns) that can be performed on data entities and associations. Transformation rules are then applied on the abstract representation based on the types of changes introduced by the data translation operators. The transformed representation is then mapped into another intermediate representation (called access path graphs) which is dictated by the external model and specific schema used for the target data base. This representation is then modified by an optimization component and used for the generation of target programs. This work stresses that the semantics of both the source and target data base be made explicit to the conversion system and be used as a basis for application program analysis and transformation. The conversion methodology described is for program conversion to account for data conversion at the logical level as well as the DBMS level.

Housel extends the work on application migration undertaken at the IBM San Jose Laboratory. This work uses a common language for specifying the abstract representation of source programs as well as for specifying the data translation operations. The language is a subset of CONVERT with some of Codd's relational operators [GG4]. The operators of the language are designed to have a simple semantics and convenient algebraic properties to facilitate program transformation. They are designed to handle data manipulation in a general hierarchical structure called a "form" as well as relational tables. In this system,

program transformation is dictated by the data mapping operations applied to the source data base. The proposed model assumes that the inverse of these data mapping operators exists; i.e., the source data base can be reconstructed from the target data base by applying some inverse operators on the target data base. More precisely, it is assumed that M'(T) = S where S is the source data base, T is the target data base, and M is the mapping function. Thus, program conversion is done by substituting the inverse M'(T) into the specification language statements (the abstract representation of the source program) for each reference to the source data base. This process is followed by a simplification procedure to simplify the resulting statements (the target abstract representation of the program). The author points out that the assumption on the existence of M'(T) restricts the scope of the conversion problem handled by the proposed approach.

Presently, the Data Base Program Conversion Task Group (DPCTG) of the CODASYL Systems Committee is investigating the application group conversion problem. The group is looking into various aspects of the problems including decompilation of COBOL application programs, semantic changes of data bases and their effects on application programs, program conversion techniques and methodologies, etc.

To this date, the work on application program conversion is still very much at the research stage and more progress has to be made before we can start actual implementation. The problems of automatic application program conversion are multitudinous and extremely complex. Current research indicates that program conversion is possible for some types of data conversion, but the complexity of program conversion depends on how drastically the data has been modified. Further research needs to be undertaken to determine what can be done automatically, what can be done semi-automatically, and what cannot be done at all. A fully automatic tool is hard to achieve. Building semi-automatic systems or systems which provide aids for manual conversion would be a more realistic goal.

Current Research Directions. The current research has uncovered several problems which need to be investigated further before the implementation of a generalized conversion tool can be attempted. The following issues are believed to be important for future research:

Semantic Description of Data Base and Application Programs Based on the work by Su and Liu [PT13] and the study of the DPCTG group, it is quite clear that a program conversion system would need more information about the

semantic properties of the source and target data bases than the information provided by the schemas of the existing DBMS. Semantic information of the data bases is an important source for determining the semantics of application programs which is the real bottleneck of the application program conversion problem. Future research needs to be conducted to 1) model and describe the semantics of application programs, 2) study the meaningful semantic changes to data bases and their effect on application programs, and 3) derive transformation rules for program conversion which account for the meaningful changes. Several existing works on data base semantics [M11,SM1,2,3,DL29] may provide a good basis for future works on this subject.

Equivalency of Source and Target Programs Data conversion may alter the semantic contents of the source data base. A converted application program may or may not perform the identical operation on the target data as the source program on the source data. For example, it may not retrieve, delete, or update the same data as the source program because some records may be deleted and data relations may have been changed in data conversion. It is not clear at all how we can prove, in general, that a target program generated by a conversion system still preserves the original intent of the source program. Naturally, if the source data can be reconstructed from the target data without losing the original data relations and occurrences, we can establish the equivalence relation between the source and target programs based on the same effort they have on the source and target data.

Decompilation Program conversion via decompilation is a technique whereby a data base application program is first transformed into an operationally equivalent higher order language or an abstract representation and then returned to a usable language level in a converted form. The transformation to a higher order language level is a decompilation process and the process of returning the program to a language level appropriate to conventional compilers is a compilation process. The underlying concept is that the decompilation to a higher order language can produce a functionally equivalent program that does not contain the DBMS, data model and data dependencies that inhibit the conversion process. That is, the decompiled program has the same "intent" while being unrelated to the changed DBMS environmental conditions. The changed environmental conditions should be easily incorporated into the program during the process of compiling the program back into a form appropriate to the new system.

Some researchers think that this would be the preferred method to effect DML/host language program conversion. It should avoid many of the efficiency/restriction drawbacks inherent in current automated methods, while being more cost effective and less error prone than current manual methods (e.g., program rewrite).

One likely disadvantage to this method is that in order to use it to convert existing data base application programs the programs may have to first be manually altered to place DML related code in a structured format. This disadvantage is to be expected because of the ambiguity inherent in the organization of DML/host language programs. However, the development of structured programming templates designed for DML related code should provide a means for creating programs that are convertible by the decompilation method. Structured templates might also provide the needed insight toward the development/selection of an appropriate high level language into which programs can be compiled. Some initial concepts of data base program templates have been proposed by the University of Michigan [PT10].

Conversion Aids A system which provides assistance to conversion analysts would seem to be a practical tool and a feasible task. Given the information about data changes and semantics of the data, a system can be built to analyze application programs to 1) identify and isolate program segments which are affected by the data changes, 2) detect inefficient code in the programs, 3) produce a program execution profile [GG1] which gives an estimate of the computation time required at different parts of the program, and 4) detect, in some cases, the program code which depends on the programmer's assumption of data values, ordering of records, record or file size, etc. The data obtained in 1, together with some on-line editing and debugging aids, would speed up the manual conversion process. The data obtained in 2 and 3 would be useful for producing more efficient target programs and the data obtained in 4 would help the conversion analyst to eliminate the "implicit semantics" in programs which makes the program conversion task (manual or automatic) extremely difficult. A more complete cross-referencing than that usually produced by today's compilers can assist the conversion analysts in identifying ramifications of changes to programs. An example of such a product is the Data Correlation and Documentation system produced by PSI-TRAN Corporation. One technique, sometimes used during a conversion process that has been initiated by a data base structure change, is to alter the names of effected data base items in the DDL only and use errors generated by the compiler to locate those program segments needing changes. A more complete cross-referencing system would be a much better tool, if it were available.

Optimization of Target Program As the result of data conversion, multiple access paths to the same data may occur. This is because redundant data may be introduced or new access paths may be added in the course of data conversion. In this situation, a conversion system will have the choice of selecting a path to generate the target program. The efficiency of the program during execution time may depend on the selection of optimized access path during program conversion. Also, for reasons of achieving generality, some program conversion techniques proposed [PT13,14] convert small segments of programs or the equivalent of DML statements separately. It is necessary to do a global optimization or simplification to improve the converted program. Techniques for program optimization related to program conversion need to be investigated.

6.2.3 Prototype Conversion Systems Analysis. This section analyzes the state of the art of generalized data conversion systems. It summarizes what has been learned in the various prototypes. The prototypes have yielded encouraging results, but some weak points have also emerged. A section below lists some questions that remain to be answered and comments on additional features that will be necessary to enhance usability. The following section on Analysis of Architecture analyzes some implementation issues which can affect the cases where a generalized conversion system can be applied.

Where Do We Stand. The prototype systems described in Section 6.2.2 have been used in a few conversions. While some of these tests were made on "toy files," a few of the tests involved data volumes from which realistic performance estimates can be extrapolated. This section will summarize the major tests that were done with each of the prototypes.

The Penn Translator The translator developed by Ramirez at the University of Pennsylvania [DT3,6] processes single sequential files to produce single sequential target files. Facilities exist for redefining the structure of source file records, reformatting and converting accordingly. Conversion of the file can be done selectively using user-defined selection criteria. Block size, record size, and character code set can be changed, and some useful data manipulation can be included.

The translator was used in several test runs on an IBM/370 Model 165. The DDL to generated PL/1 code expansion ratio was 1:4, so coding time was reduced.

A further test of the Penn Translator was conducted by Winters and Dickey [TA1]. An experiment was conducted comparing a conventional conversion effort against the Penn Translator (slightly modified). Two source files stored on IBM 1301 disks under a system written for the IBM 7080 using the AUTOCODER language were converted to two target files suitable for loading into IMS/VS. Much of the data was modified from one internal coding scheme to another. The conversion required changing multiple source files to multiple target files.

The conventional conversion took seven months versus five months for the generalized approach, a productivity improvement of roughly thirty percent. Time for adapting the translator, learning the DDL, and adapting to a new operating system is included in the five month figure. Without these, an estimate of three months was made for the conversion using the generalized approach.

The SDC Translator The translator described in [R3,4] was implemented during 1975-1976. The translator could handle single, hierarchical files from any of three local systems--TDMS, a hierarchical system which fully inverts files; DS/2, a system which partially inverts files; and ORBIT, a bibliographic system which maintains keys and abstracts. Data Bases were converted from TDMS to ORBIT, from TDMS to DS/2, and vice-versa, and from sequential files to ORBIT. TDMS files were unloaded using an unload utility. Target data bases were loaded by target system load utilities.

The total effort for design and implementation was about three man-years. The system was implemented in assembly language on an IBM/370 Model 168, and occupied about 40 K-bytes, not including buffer space which could be varied. The largest file tested was on the order of 5 million characters and the total conversion time was about 1 minute of CPU time per 2.5 megabytes of data.

The work was discontinued in 1976.

The Honeywell Translator The prototype file translator developed at Honeywell by Bakkom and Behymer [DT11] performed file conversions (one file to one file) among files from IBM, Honeywell 6000, Honeywell 2000, Honeywell 8200 sequential and indexed sequential files. Data types of fields could be changed as well as field justification and alignment. New fields could be added to a record and fields could be permuted within a record. File record format (fixed, variable, blocked, etc.) could be changed and a compare utility was available for checking the consistency of files with different field organizations and encodings.

Tests of up to 10,000 records were run. Performance of 15 milliseconds per record was typical (Honeywell Series 6000 Model 6080 computer). The prototype has been used in a conversion/benchmark environment but has not been offered commercially.

The Michigan Translator Version IIB, Release 1.1 of the Michigan Translator was completed for the Defense Communications Agency in October 1977 [UT16]. It offers complete conversion/restructuring facilities for users of Honeywell sequential, ISP, or I-D-S/I files. Up to five source data bases of any type may be merged, restructured or otherwise reorganized into as many as five target data bases, all within a single translation. Data Base description is accomplished by minor extensions to existing I-D-S DDL statements. Restructuring specification is easily indicated via a high level language. Tests performed to date included a conversion of a 150,000 record I-D-S/I data base with a total elapsed time of 24 hours (500 milliseconds per record). A given translation can be broken off at any point to permit efficient utilization of limited resources and also protect against system failures. The user is provided with the capability of monitoring translation progress in real time.

XPRS Test cases with the XPRS system have focussed on functionally duplicating earlier real conversions done by conventional methods. Several cases have been programmed. Each case involved at least two input files. Generally, there was a requirement to select some instances from one file, match with instances in another file, eliminate some redundant or unwanted data, and build up a new hierarchical structure in the output. In several cases there was a need for conditional actions based on flags within the data. In all cases, the XPRS languages were found to be functionally adequate to replicate the conversion. A productivity gain of at least fifty percent in total analysis, coding, and debugging time was achieved. Test runs were conducted on several thousand records. Performance was deemed adequate in that XPRS can restructure data at least as fast as it can be delivered from direct access storage. No detailed performance comparisons were made comparing XPRS-generated programs with custom written programs.

Questions Remaining To Be Answered. Given that several prototype data translation systems are operational in a laboratory environment, there is a little question concerning the technical feasibility of building generalized systems. The remaining questions pertain to the use of a generalized system in "real world" data conversions involving a wide variety of data structures, very large data volumes, and significant numbers of people. Three major

questions to be resolved are:

1. Are the generalized systems functionally complete enough to be used in real conversions, and if not, what will it take to make them functionally complete?

2. Can the people involved in data conversions use the languages? What additional features are necessary to enhance usability?

3. Overall, what is the productivity gain available with the generalized approach?

Within the next year, prototype systems will be exercised on a variety of real-world problems in data translation, and concrete answers to these questions should be available. The systems being further tested for cost-effectiveness are the Michigan Data Translator, the IBM XPRS system, and the Bell Laboratories ADAPT system.

To date, preliminary results have been promising. A significant sample size on which to do analysis of productivity gain should be available at the end of the year of testing.

A number of factors must be taken into account in measuring the cost-effectiveness of the generalized data translator versus the conventional conversion approach. These factors include:

- ease of learning and using the higher level languages which drive the generalized translators;
- availability of functional capability to accomplish real-world data conversion applications within the generalized translators;
- overall machine efficiency;
- correctness of results from the conversion;
- ability to respond in timely fashion to changes in conversion requirements (conversion program "maintenance");
- debugging costs;

- ability to provide "bridge back" of converted data to old applications;

- ability to provide verification of correctness of data conversion;

- capabilities for detection and control of data errors.

The languages used to drive generalized data translators are high-level and non-procedural; they provide a "user-friendly" interface to the translators. Since the languages are high-level, programs written in them have a better chance of being correct. Experience to date with DEFINE and CONVERT, the languages which drive XPRS, has shown that users can learn these languages within a week; it has also shown that some practice is necessary before users start thinking about their conversion problem in non-procedural rather than procedural terms.

In early test cases, the languages which drive generalized data translators have been found to be functionally adequate for many common cases. In those cases lacking a feature, a "user hook" facility is often provided. However, forcing a user to revert to a programming language "hook" defeats the purpose of the high level approach, and interfacing the hook to the system requires at least some knowledge of system interfaces. Thus, high level languages must cover the vast majority of the cases in order to succeed; otherwise, users will perceive little difference over conventional approaches.

Facilities for detecting and controlling data errors in the generalized systems are very important, and most of the prototypes do not yet do a complete job in this area. However, the generalized packages offer an opportunity for generalized, high level methods for dealing with data errors during conversion, and it could well be that once these error packages are developed, they will contribute to even larger productivity gains than have been experienced to date.

The high-level language approach to driving generalized translators should provide the ability to respond to changes in conversion requirements with relative ease. Since large conversions often take one or more years, it is not unusual for the target data base design to change or for new requirements to be placed on the conversion system. In other words, in a large conversion effort, the programs are not as "one shot" as is commonly believed. In large conversions, the savings in conversion program maintenance could be significant.

Generalized systems can also be used to map target data back to the old source data form, assuming the original conversion was information-preserving. This capability provides a means for verifying the correctness of the data conversion. In addition, this capability can be used as a "bridge back" to allow users to continue to run programs which have not yet been converted against the data in the old format. Using a generalized system in this way allows phased conversion of programs without impacting user needs during the conversion period.

In an environment where a generalized translator is used regularly as a tool for conversion, costs associated with the debugging phase should be decreased. Common, debugged functional capabilities will be utilized, whereas it is unusual in the conventional approach for common conversion modules to be developed. Thus, each new conversion system requires debugging.

Usability The usability of generalized data translation systems must also be evaluated. Experience to date indicates that the languages are easy to learn and use. However, it would be wrong to think that these prototypes are mature software products or that they can be used in all conversions. This section discusses some of the unanswered questions with respect to usability of the current data conversion systems.

One question concerns the level of users of the generalized languages. Current prototypes have been used by application specialists and/or members of a data base support group. The systems have not yet been used by programmers, and the question remains whether programmers (as opposed to more senior application specialists and analysts) will be able to use the systems productively. There is no negative data on this point; the systems have not been used widely enough.

At present, all the systems require a user to describe explicitly the source data to be accessed by the read step using a special data description language. These data description languages are generally easy to learn and use; they resemble statements in the COBOL Data Division. However, the writing of the description is a manual process which can be tedious because a person may have to describe a file with hundreds of fields. Ideally, a data conversion system should be able to make use of an existing data description, such as those existing in a data dictionary or a system·COBOL macro library. As evidenced by the Michigan Data Translator [DT16], it is reasonable to expect that such an interface will be available as data conversion systems evolve. Note, however, that a data dictionary or COBOL

macro library link may not necessarily solve the problem. Data in current systems is not always fully enough defined to be converted. This is especially true with non-data base files. In these files, data definition often is embedded in the record structures of the programs, and a full definition depends on a knowledge of the procedural program logic. Even with existing data bases, some fields and associations may not be fully defined within the system data base description. Thus, the user can expect a certain amount of manual effort in developing data definitions. If existing documentation is incomplete, this can be a time consuming task, though it probably must be done regardless of whether a generalized package is used or not.

Another area where a user may have to expend effort is in the unload step of the data conversion process. The data description languages used to drive the read step have a limited ability to deal with data at a level close to the hardware (e.g., pointers, storage allocation bit maps, etc.). Generally one assumes that a system utility program can be used to unload source data and remove the more complex internal structures. Another alternative is to run the read step on top of an existing access method or data base management system with the accessing software removing the more complex, machine dependent structures. While acceptable alternatives in a great many environments, including most COBOL environments, some cases may exist where neither approach will work. For example, a load/unload utility may not exist, or a file with embedded pointers which was accessed directly by an assembly language program might not be under the control of an access method. For these cases, the user is faced with complexity during the unload step. The complexity associated with accessing the data would appear to be a factor for either the conventional methods or for the generalized approach. However, in cases such as those above, some special purpose software may have to be developed. Note that some research [SL8] has examined the difficulty of extending data description languages to deal directly with these more complex cases and has concluded that providing the data description language with capabilities to deal with more complex data structures greatly complicates the implementation and has an adverse affect on usability. Thus, special purpose unload programs will continue to be required to deal with some files.

Analysis of Architectures. This section discusses some of the different approaches that have been taken in implementing the prototype data conversion systems. The objective is to analyze some of the performance and usability issues raised by the prototypes.

Two approaches have been used in the prototypes--a generative approach in the Penn Translator and XPRS, and an interpretive approach in the Michigan Data Translator. In the generative approach, a description of the input files, output files, and restructuring operations is fed to a program generator. From these descriptions, special purpose programs are generated to accomplish the described conversion. In both the Penn Translator and XPRS, PL/1 is the target language for the generator. The generated PL/1 programs are then compiled and run. In the interpretive approach, tables are built from the data and/or restructuring description. These tables are then interpreted to carry out the data conversion.

In data conversion systems, as in other software, an implementation based on interpretation can be expected to run considerably more slowly than one based on generation and compilation. Initial experience with prototype data translators has shown that there is much repetitive work, strategies for which can be decided at program compilation/generation time. Also, there is a good deal of low level data handling, such as item type conversions. Thus, those implementations based largely on an interpretive approach run more slowly, and the ability to vary bindings at run time does not appear to be necessary. Interpretation was chosen in the prototypes for ease of implementation, and in the future it can be expected that a compilation-based approach or a mixture of compilation with interpretation will be the dominant implementation architecture. However, for medium scale data bases, the machine requirements of the interpretive data conversion prototypes are not unreasonable, and overall productivity gains are still possible.

Performance measurements with conversion systems based on the generative approach indicate that generalized systems can be quite competitive with customized programs. In one case, the program generated by the data conversion system ran slightly faster than a "customized" program which had been written to do the same job. However, this example could well be the exception and it would be naive to expect this in general. The reason generalized packages can be competitive is that they often have internal algorithms which can plan access strategies to minimize I/O transfer and/or multiple passes over the source data. "Customized" conversion programs written in a conventional programming language often are not carefully optimized, since the expectation is that the programs will be discarded when the conversion is done.

A second architectural difference involves the use of an underlying DBMS or not. In both the Penn Translator and XPRS, the generated PL/1 program, then executing, accesses sequential files, performs the restructuring, and writes sequential files. On the other hand, the Michigan Data Translator functions as an application program running on a network structured data base management system. Thus, the interpreter makes calls to the underlying DBMS to retrieve data during restructuring and puts restructured data into the new data base.

The two approaches offer different tradeoffs. For example, the Michigan Data Translator can make use of the existing extraction capabilities of a DBMS and perform partial translations easily. In addition, since it operates directly within the network data model, a user does not have to think of "unloading" data to a file model and then reloading it back; rather, the user describes a network to network restructuring much more directly.

On the other hand, when converting non-data base data to a data base, the use of an underlying DBMS as part of a data translator implies a second order data conversion problem--the non-data base data must be converted into the DBMS of the data conversion system, which may or may not be difficult. It can be difficult, for example, when the data model of the data being converted differs significantly from the data model of the DBMS upon which the conversion system is based. Also, the use of an underlying DBMS may also require more on-line storage, whereas the file oriented conversion systems can be made to run tape-to-tape. This can be important in very large data base conversions.

In the future, one can expect that data conversion systems will offer a variety of interfaces to accommodate various kinds of conversion situations. For example, it is possible to interface the "file-oriented" conversion systems to run as application programs on top of existing data base management systems. It is also possible to develop "reader programs" to load non-data base data into conversion systems based on a DBMS. In addition, more automated interfaces to data dictionary packages can be expected in order to improve usability and obviate the need for multiple data definitions.

One possible performance problem with generalized conversion systems lies in the unload phase. For reasons of usability, generalized conversion systems usually rely on an unload utility program to access the source data, thus isolating the conversion package from highly system specific data. A potential problem with this approach is that the unload package may not make good use of existing access

paths or may tend to access the source data in a fashion which assumes that the data has recently been reorganized (with respect to overflow areas, etc.). In cases where the data is badly disorganized, a customized unload program which accessed the data at a lower level might run considerably faster, and for very large data bases might be the only feasible way to unload the data. It is not clear how common this case is, and one can usually make the argument that the "special" unload software could be interfaced to the generalized package. However, from a practical standpoint, the unloading phase on a very large, badly disorganized data base is a performance unknown, and more sophisticated unload utilities may have to be developed as part of the generalized packages.

Summary Detailed performance and productivity figures for major conversions should be available in about one year. Expectations are that machine efficiency of the generalized packages based on a generation/compilation approach will be acceptable (no worse than a factor of 2) when compared with conventional conversion programs. Additional enhancements to improve usability can be expected, especially in the areas of data error detection and control and interfaces to data dictionary software. If the savings in conversion program analysis and coding times--often fifty percent or more--are confirmed, then the generalized conversion systems will be ready for extensive use.

6.3 OTHER FACTORS AFFECTING CONVERSION

In this section we look at the conversion problem from two aspects. First, we address the question--What can we do today to lessen the impact of a future conversion? Second, we look to the future to see what effects future technology and standards will have on the conversion process.

6.3.1 Lessening the Conversion Effort. In order to identify guidelines for both reducing the need for conversion and for simplifying conversions which are required, one must consider the entire application software development cycle because poor application design, poor logical data base design, inadequate use of and inappropriate DBMS selection could each lead to an environment which may prematurely require an application upgrade or redesign. This redesign could, in many cases, require a major data base conversion effort.

The set of guidelines specified below is not intended as a panacea. Instead, it is meant to make designers aware of strategies which make intelligent use of current technology. It is doubtful that all conversions could be avoided if a project adhered strictly to these proposed guideliines. However, adherence to the principles set forth by these guidelines could certainly reduce the probability of conversion and, more importantly, simplify the conversions that are required.

With respect to application design and implementation, the more the application is shielded from system software and hardware implementation details, the easier it becomes for a conversion to take place. For example, a good sequential access method hides the difference between tapes, disks, and drums from the application programs which use the access method.

The logical data base design should be specified with a clear understanding of the information environment. A good logical data base design reduces the need to restructure because it actually models the environment it is meant to serve. Introduction of data dependencies in the data structure should, if possible, be kept to a minimum. An analysis of the tradeoffs between system performance and likelihood of conversion should definitely be made.

Selecting the wrong or non-optimal data base management system, given the application requirements, is also a key problem which can lead to unnecessary and large conversion efforts. The prospective user of a DBMS should, for example, carefully evaluate the data independence characteristics of a proposed DBMS.

The underlying principle of the guidelines which follow is that decisions can be made at the system design and implementation stages which are crucial to the stability of the applications.

Application Design Guidelines.

Requirements Analysis Many of the decisions made during the requirements analysis stage of system development affect the long-term effectiveness of the application system (data base design as well as application programs). Questions such as what functions does the application require, who will the data base serve and how will they use the data base, what are the possible future uses of the data, and what are the performance constraints of the application are answered at this stage. It is essential that the designer understand the information environment as much as possible at the outset in order to lessen the probability that

frequent conversions will be necessary.

Requirements analysis should focus on information needs and should minimize constraints being imposed by the physical environment since it can distort the designer's view of the application system's true objectives. The influence of the physical environment should be considered secondarily, in order that the designer be fully aware of the resulting compromises to the logical requirements. This is not intended to imply that consideration of the physical environment is unimportant. Indeed, if the physical environment is ignored, the effect could be development of a set of requirements that are impossible to meet within existing physical and cost constraints.

Program Design Guidelines Three underlying principles motivate this discussion of application program design. They are:

- design for maintainability
- design for the application
- data independence

Keeping sight of all of these during the design of the application program will lessen conversion effects by rendering the application as free as possible from physical considerations.

Designing for maintainability implies that the application should be written in a high-level language with a syntax that permits good program structure. Structured programming techniques such as top-down program design and implementation should be used throughout. The system should be modular with relatively small, functionally oriented programs. The programs should all be well commented and organized for readability. Design reviews and program walkthroughs also help to expose errors in the overall design and "holes" in the application logic at an early stage. It has been well documented that these steps help ease making program modifications.

One error which is often made in designing programs in a DBMS environment is to let the capabilities of the DBMS drive the design rather than the application. This design error can yield programs which are unnecessarily dependent upon the features of a specific DBMS. For example, in System 2000 one can use a tree to represent a many-to-many relationship instead of using the LINK feature. The parent/child dichotomy that results is an efficient but

arbitrary contrivance that cannot easily be undone later on. The key principle here is to concentrate on what results are desired rather than on the implementation details of achieving these results. Simplicity and generalization of the design will provide a very high level of interface to the application programmer which will, in turn, minimize the total amount of software, provide the greatest degree of portability, maintainability, devide independence, and data independence.

Of extreme importance in program design is the notion of data independence; i.e., insulating the application program from the way the data is physically stored.

Layered Design

In the area of application design, the motivating factor for mitigating the effects of a conversion is to insulate logical operations from physical operations. One of the concepts applied to achieve this is layered design. That is, designing the application as a series of layers, each of which communicates with the system at a different level of abstraction. One can visualize this as an "onion," with hardware as its core and layers of successively more sophisticated software at the outer layers. The user interacts with the outermost skin of the onion, at the highest level of abstraction.

If application programs are written at the outermost layers of the onion, then these programs are smaller, easier to understand and, therefore, easier to modify or convert than programs written at lower layers. For example, introduction of a new mainframe will require conversion of the software which references the particulars of the mainframe. However, since the layers are constructed so that physical machine and device independence is realized above some level, only the software below that level is subject to modification. To the extent that application programs stay at the outermost layers (i.e., above the critical layer) reduced conversion effects can be achieved.

We can thus summarize the goal of program design as follows:

- to provide the highest possible application interface to the program
- to maximize program independence from the characteristics of the mainframe, peripherals, and data base organization

- to maximize portability of the application program through the use of high-level languages

- to maintain a clean program/data interface

Programming Techniques

The previous sections of this chapter have focused on the design decisions which should be made to alleviate the conversion problem. However, regardless of how noble these goals are, poor implementation decisions can go a long way towards diminishing the returns of a good design. Equally important to intelligent design is a set of programming techniques and standards which prohibit programmers from introducing dependencies in code. For example, a "clever" programmer may introduce a word size dependency in a program by using right and left shifts to effect multiplication and division. Of course, there are no hard and fast safeguards against using tricky coding techniques; an effort must be made to make the programmer conscious of the consequences of this kind of coding. In particular, a programmer should not be allowed to jump across layers of the onion, such as using an access method to read or write directly data bases.

Data Base Design. Perhaps the most costly mistake a designer can make is an error in the data base design because it has a direct effect on the information that is derivable and the application programs that are created. Incorrect or unanticipated requirements can lead either to information deficient data bases or overly complex and general design. An inadequate logical design has the potential for complex user interfaces or extremely long access time. A poor physical design can lead to high maintenace and performance costs. Unfortunately, data base design is still an art at the present time. Two surveys report the results in the area to date. Novak and Fry [DL26] survey the current logical data base design methodology and Chen and Yao [DL34] review data base design in general. The work of Bubenko [DL31] in the development of the CADIS system and the abstraction and generalization techniques of Smith and Smith [DL29,30] show promise.

An accurate logical design can still be unnecessarily data dependent. Dependencies are inadvertently or deliberately introduced in the interest of improving system performance. In essence, "purity" is compromised to gain processing efficiences. Since optimization is a worthwhile goal, insisting on absolute purity may be unreasonable. However, the data base designer should at least be aware of contrivances and, therefore, be in a position to evaluate the relative effects a design decision may have. Designers should become sensitive to their decisions by asking: "How

will the data model be affected by a future change in performance requirements? Have I done a reasonable job in insulating applications from data structure elements that are motivated strictly by performance considerations?"

Some examples of induced data dependencies in logical data base design which may impact upon conversion are:

- The use of owner-coupled sets in DBTG to implement performance-oriented index structures or orderings on records.
- Storing physical pointers (or data base keys) in an information field of a record
- Combining segment types (in DL/1) to reduce the amount of I/O required to traverse a data base.

DBMS Utilization and Selection. Selection of a DBMS can have a major impact on conversion requirements. Of importance in evaluating a DBMS is to consider products exhibiting the highest level user interface.

A high level DBMS is characterized by both a powerful set of functions and a high degree of data independence from the point of view of the application. With respect to functions, that is, the DML, the distinction between "high level" and "low level" has traditionally centered on whether the DBMS provides user operations on sets of records (select, retrieve, update, or summarize all the records or tuples which satisfy some conditions) or whether one is restricted to record-at-a-time processing ("navigation"). The DBMS with the "high-level" set operation approach is significantly more desirable than the navigational record-by-record approach.

DBMS prospects should evaluate the data independence characteristics of a proposed product. Systems are preferred which support an "external schema" or "subschema" feature which permits the record image in the application program (the user work area) to differ significantly from the data base format. However, the subschema concept is only one aspect of data independence. In general, it is necessary to determine in what ways and to what extent the application interface is insulated from performance or internal format options. For instance, will programs have to be modified if:

- a decision is made to add or delete an index?

- the amount of space allocated to an item is increased or decreased?

- chains are replaced by pointer arrays?

Other conversion related questions about DBMS products include the following:

- Are there adequate performance and formatting alternatives? Are there too many (i.e., unproductive or incomprehensible) tuning options? Are there adequate performance measurement techniques and tools to guide the exercise of these choices?

- Does the system automatically convert a populated data base when a new format option is selected?

- Aside from tuning, does the DBMS gracefully accommodate at least simple external changes such as adding or deleting a record or item type?

- Are there other useful high level facilities associated with or based on the DBMS, such as a report writer, query processor, data dictionary, transaction monitor, accounting system, payroll system, etc.?

- Is there a utility for translating the data base into an "interchange form;" i.e., a machine independent, serial stream of characters?

- Is the vendor committed to maintaining the product across new operating system and hardware releases/upgrades? Conversely, is the vendor prepared to support the product in order released of the operating system, so the user will not be forced to upgrade?

- What hardware environments are currently supported and what is the vendor's policy regarding conversion to another manufacturer's mainframe?

- What programming language interfaces are available? Can the same DBMS features be used if there is a migration, say, from COBOL to PL/1?

- How intelligent is the system's technique for organizing data on the media? Specifically, will performance deteriorate at an inordinate rate as updating proceeds? How often will reorganization (cleanup) be required? Does the DBMS have a built-in reorganization utility? How does the user determine the optimal time to reorganize?

- Are the language facilities and data modeling facilities of DBMS adequate for the anticipated long term requirements of the enterprise? What is the risk of having to convert to a new DBMS?

- Likewise, are the performance characteristics and internal storage structure limitations adequate to meet the long term requirements (response times, data base sizes) of the enterprise?

- Are there facilities to assist the user in converting data from a non-DBMS environment or from another DBMS? For instance, can a data base be loaded from one or more user defined files?

6.3.2 Future Technologies/Standards Impact.

In this section we discuss trends in computer hardware technologies, DBMS software directions, and standards development, and consider their impact on data and program conversion. We intend to make the reader aware of what to expect in terms of conversion problems rather than give a complete assessment of future technologies. Therefore, we discuss only technologies and standards that will impact conversion problems.

The first three parts discuss the areas of hardware, software, and standards and their impact on conversion in some detail. The last part summarizes the major points of our assessment without going into detailed reasoning.

Hardware and Architectural Technologies. The cost and performance of processor logic and memory continue to improve at a fast rate. As a result, overhead costs are more acceptable, especially when such costs save people's time and work, and provide user oriented functions that do not require a computer expert. In particular, one can now think about using generalized conversion tools not only when it is required as a result of hardware or software changes, but also as a result of a changing application that requires a new more efficient data base organization. What could have been a prohibitive cost for a data base conversion in the past, may not be a major factor in the future.

At the same time, the cost/performance improvement contributes to the proliferation of data bases and therefore accentuates the need of generalized conversion tools. The more cost effective is the process of accessing and maintaining data, the more data is collected on computers. Improvements in hardware (as well as software) technologies create more need for data and program conversion. In addition, the emergence of new technologies, such as communication networks, add another level of sophistication to the way that data can be organized and used. Distributed data bases, where multiple data bases (or subsets of data bases) may reside on different machines, require tools for the integration and the correlation of data. Invariably, data will need to move from system to system dynamically, possibly moving between different hardware/software systems. In this environment, generalized tools for dynamic conversion will become a necessity.

In recent years, two promising approaches to data management hardware technologies have been pursued. One is the specialized data management machine and the other is the backend data management machine. As will be explained next, both approaches can help simplify the conversion problem.

The specialized data management hardware is based on the idea of using some kind of an associative memory device, a device that can perform a parallel access to the data based on its content. Such a device eliminates the necessity for organizing the internal structure of a data base using indexes, hash tables, pointer structures, etc., which are primarily used for fast access. As a result, the data can be essentially stored in its external logical form, and the data management system can use a high level language based on the logical data structure only. The conversion process is simplified since data is readily available in its logical organization. Referring to the terminology used in previous sections, the functions of unloading and loading of the data base can be greatly simplified. Also, no restructuring will be required because of a change in data base use, since the physical data base organization can be to a large degree independent of its intended use. In addition, the program conversion problem is simplified as a result of the program interfacing to the DBMS using a high level logical language.

Similar benefits can be achieved if backend machines are used. A backend machine is a special processor dedicated to managing storage and data base on behalf of a host computer. The primary motive for the backend machine is to off-load the data management function from the host to a specialized machine that can execute this function at much lower cost. From a conversion standpoint, the separation of

data management functions from the host promotes the need for a high level logical interface that provides the advantages discussed above. Another advantage is that it is possible to migrate from one host machine to another without affecting the data bases and their management, alleviating the need for data conversion if the same backend machine is used with the new host.

Mass storage devices, such as the video disks, make storing very large data bases, in the order of 10 to the 10th power characters, cost effective. Converting large data bases of this size compounds the cost considerations merely by the processing of this large amount of data. As a result, such data bases will tend to stay in the same environment for longer periods of time. The use of specialized data management machines or dedicated backend machines in conjunction with these mass storage devices can help postpone the need for data base conversion.

Finally, we should mention the growing use of minicomputers in supporting data management functions. DBMSs now exist on many minicomputers, with more forthcoming. The proliferation of minicomputers which support data bases can only increase the needs for generalized conversion tools.

Software Development Trends. Much of the work over the last years in the data management area has concentrated on techniques that clearly separate the logical structure of the data base from its physical organization. This concept, called "data independence" was introduced to emphasize that users need not be exposed to the details of the physical organizations of the data base, but only to its logical relationships. This led to the development of data access and manipulation languages that depend on the logical data model only. The effect of this trend is similar to that of using specialized data management machines and backend machines discussed previously; namely, the simplification of the unload and load functions since the interface to the DBMS is provided at the logical level only, and the simplification in program conversion for similar reasons.

At the user end of the spectrum, it seems reasonable to assume that the diversity of data models (network, relational, hierarchies and other views that may be developed in the future) will be required for many more decades. This is especially true since there are problem areas that seem to map more naturally into a certain model. Furthermore, it is often the case that users do not agree on the same model for a given problem area. Obviously, this state of affairs only accentuates the need to generalize conversion tools that can restructure data bases from one

model to another. Even with the development of large scale associative memories, data structures will likely provide economic rationales for their contrived use. Another possibility is the use of a common underlying data model that can accommodate any of the user views. However, this approach will still require some type of a dynamic conversion process between the common view and each of the possible user views.

Standards Development. There is much work and controversy in developing standards for DBMS. Standards that are oriented to determine the nature of the DBMS are hard to bring about even in a highly controlled environment because of previous investment in application software and data base development, and because of disagreement. For example, there is still much controversy whether the network model proposed by the CODASYL committee is a proper one. It seems reasonable to assume that there will always be non-standard DBMSs. Further, even if such a standard can be adopted, different DBMS implementations will still exist, resulting in different physical data bases for the same logical data base. In addition, one can safely assume that restructuring because of application needs will still be necessary, and that changes in the standard itself may require conversion.

A standard that is more likely to be accepted is one that affects only the way of interfacing to a DBMS. In particular, from a conversion standpoint, a standard interchange data form (SIDF) will be most useful. A SIDF is a format not unlike a load format for DBMSs. Any advanced DBMS has a load utility that requires sequential data stream in a pre-specified format. If a standard for this format can be agreed upon, and if all DBMSs can load and unload from and to this format, then the need for reformatting (as described earlier) is eliminated. The conversion process can be reduced to essentially restructuring only, given that unload and load are part of the DBMS function. A preliminary proposal for such a standard was developed by the ERDA Inter-Working Group on Data Exchange (IWGDE) [GG2]. However, it is only designed to accommodate hierarchical structures. Consideration is now being given to the extension of the standard to accommodate more general structures (i.e., networks and relations). We believe that there are no technical barriers to the development of a SIDF, and that putting such a standard to use would alleviate a major part of the data conversion process.

Summary. The rationale for the points summarized below appear in the previous parts of this section. We will only state here our assessment of the impact on conversion problems.

- Hardware development will increase the need for generalized conversion tools (in particular, proliferation of minicomputers, computer networks, and mass storage devices).

- The reduction in hardware costs will make conversion costs more acceptable.

- Special hardware DBMS machines will simplify the conversion process (in particular, for load, unload functions, and program conversion) because they promote interfacing at the logical level.

- Software advances will not eliminate the need for conversion but can simplify the conversion process in a way similar to DBMS machines.

- Multiplicity of logical models is likely to exist, thus adding to the need of conversion tools between models.

- Standards will not eliminate the conversion problem. Even with a standard, the implementations would be different and non-standard DBMS will likely exist.

- Standards can greatly simplify the conversion problem. In particular, a standard for interchange data form will simplify load and unload function and eliminate reformatting.

What can we expect in the next five years and beyond in the data base conversion area? The state of the art has advanced enough to give hope for generalized tools. Within the next five years we can expect more generalized conversion systems to become operational but some additional work would be required for moving them from one environment to another. We can expect to have a standard form developed and agreed upon. It will probably take longer before manufacturers will see the benefit of adopting a standard form and provide load and unload facilities using it. However, we can expect them to provide some conversion tools to convert data bases from other systems to their own. It will probably take as much as ten years before the commercial availability of a generalized converter and the adherence of manufacturers to a standard interchange form. Another area of concern is the application program conversion resulting from a data base conversion. We cannot expect that a generalized solution for this problem will be achieved within the next five years. However, this problem will be simplified to a large extent as hardware and software development trends promote interfacing at the logical level.

(DL) LOGICAL DATA BASE DESIGN

DL26 NOVAK, D. and FRY, J., "The State of the Art of Logical Data base Design," Proceedings of the Fifth Texas Conference on Computing Systems, IEEE, Long Beach, 1976, pp. 30-39.

DL29 SMITH, J. M., and SMITH, D. C. P., "Data Base Abstractions: Aggregation and Generalization," ACM Transactions on Data Base Systems 2,2(1977):105-133.

DL30 SMITH, J. M., and SMITH, D. C. P., "Data Base Abstractions: Aggregation," Communications of the ACM 20,6(1977):405-13.

DL31 BUBENKO, J. A., "IAM: An Inferential Abstract Modeling Approach to Design of Conceptual Schema," Proceedings of the ACM-SIGMOD International Conference on Management of Data, ACM, N.Y., 1977, pp. 62-74.

DL34 CHEN, P. P., and YAO, S. B., "Design and Performance Tools for Data Base Systems," Proceedings of the Third International Conference on Very Large Data Bases, ACM, N.Y., 1977, pp. 3-15.

(DT) DATA TRANSLATION

DT1 FRY, J. P., FRANK, R. L., HERSHEY, E. A., III, "A Developmental Model for Translation," Proc. 1972 ACM SIGFIDET Workshop on Data Description, Access and Control, A. L. Dean (ed.), ACM, N.Y., pp. 77-106.

DT2 SMITH, D. C. P., "A Method for Data Translation Using the Stored Data and Definition Task Group Languages," Proc. of the 1972 ACM SIGFIDET Workshop on Data Description, Access and Control, ACM, N.Y., pp. 107-124.

DT3 RAMIREZ, J. A., "Automatic Generation of Data Conversion Programs Using a Data Description Language (DDL)," Ph.D. dissertation, University of Pennsylvania, 1973.

DT4 MERTEN, A. G., and FRY, J. P., "A Data Description Approach to File Translation," Proc. 1974 ACM SIGMOD Workshop on Data Description, Access and Control, ACM, N.Y., pp. 191-205.

DT5 HOUSEL, B., LUM, V., and SHU, N., "Architecture to an Interactive Migration Systems (AIMS)," Proc. 1974 ACM SIGFIDET Workshop on Data Description, Access and Control, ACM, N.Y., pp. 157-169.

DT6 RAMERIZ, J. A., RIN, N. A., and PRYWES, N. S., "Automatic Conversion of Data Conversion Programs Using a Data Description Language," Proc. 1974 ACM SIGFIDET Workshop on Data Description, Access and Control, ACM, N.Y., pp. 207-225.

DT7 FRANK, R. L., and Yamaguchi, K., "A Model for a Generalized Data Access Method," Proc. of the 1974 National Computer Conference, AFIPS Press, Montvale, N.J., pp. 437-444.

DT8 TAYLOR, R. W., "Generalized Data Structures for Data Translation," Proc. Third Texas Conference on Computing Systems, Austin, Texas, 1974, pp. 6-3-1 ff.

DT9 UNIVAC, UNIVAC 1100 Series Data File Converter, Programmer Reference UP-8070, Sperry Rand Corporation, March, 1974.

DT10 YAMAGUCHI, K., "An Approach to Data Compatibility, A Generalized Access Method," Ph.D. dissertation, The University of Michigan, 1975.

DT11 BAKKOM, D. E., and BEHYMER, J. A., "Implementation of a Prototype Generalized File Translator," Proc. 1975 ACM SIGMOD International Conf. on Management of Data, W. F. King (ed.), ACM, N.Y., pp. 99-110.

DT12 NAVATHE, S. B., and MERTEN, A. B., "Investigations into the Application of the Relation Model of Data to Data Translation," Proc. 1975 ACM SIGMOD International Conf. on Management of Data, W. F. King (ed.), ACM, N.Y., pp. 123-138.

DT13 BIRSS, E. W., and FRY, J. P., "Generalized Software for Translating Data," Proc. of the 1976 National Computer Conference, Vol. 45, AFIPS Press, Montvale, N.J., pp. 889-899.

DT14 LUM, V. Y., SHU, N. C., and HOUSEL, B. C., "A General Methodology for Data Conversion and Restructuring," IBM R & D Journal, Vol. 20, No. 5, 1976, pp. 483-497.

DT15 HOUSEL, B. C., et al., "Express: A Data Extraction, Processing, and Restructuring System," Transactions

on Data Base Systems, 2,2, ACM, N.Y., 1977, pp. 134-174.

DT16 SWARTWOUT, D. E., DEPPE, M. E., and FRY, J. P., "Operational Software for Restructuring Network Data Bases," Proc. of the 1977 National Computer Conference, Vol. 46, AFIPS Press, Montvale, N.J., pp. 499-508.

DT17 GOGUEN, N. H., and KAPLEN, M. M., "An Approach to Generalized Data Translation: The ADAPT System," Bell Telephone Laboratories Internal Report, October 5, 1977.

(GG) GENERAL

GG1 INGALS, D., "The Execution Time Profile as a Programming Tool," in Design and Optimization of Compilers, ed. by R. Rustin, Prentice Hall, 1972, pp. 108-128.

GG2 ERDA Interlaboratory Working Group for Data Exchange (IWGDE) Annual Report for Fiscal Year 1976, NTIS LBL-5329.

GG3 DATE, C. J., An Introduction Data Base System, Addison-Wesley, 1975. GG4 CODD, E. F., "Relational Completeness of Data Base Sublanguage," In Data Base Systems, Caurant Computer Science Symposia Series, Vol. 6, Prentice Hall, 1972.

(M) MODELS-THEORY

M11 CHEN, P.P.S., "The Entity-Relationship Model - Towards a Unified View of Data," Transactions on Data Base Systems 1,1(1976):9-36.

(PT) PROGRAM TRANSLATION

PT1 SHARE AD-HOC COMMITTEE ON UNIVERSAL LANGUAGES, "The Problem of Programming Communication with Changing Machines: A Proposed Solution," Comm. ACM, Aug., 1958, pp. 12-18.

PT2 SHARE AD-HOC COMMITTEE ON UNIVERSAL LANGUAGES, "The Problem of Programming Communication with Changing Machines: A Proposed Solution, Part 2," Comm. ACM, Sept., 1958, pp. 9-16.

PT3 SIBLEY, E. H., and MERTEN, A. G., "Transferability and Translation of Programs and Data," Information Systems, COINS IV, Plenum Press, N.Y., 1972, pp. 291-301.

PT4 YAMAGUCHI, K., and MERTEN, A. G., "Methodology for Transferring Programs and Data," Proc. 1974 ACM-SIGFIDET Workshop on Data Description, Access and Control, ACM, N.Y., pp. 141-156.

PT5 HOUSEL, B. C., LUM, V. Y., and SHU, N., "Architecture to an Interactive Migration System (AIMS)," Proc. 1974 ACM-SIGFIDET Workshop on Data Description, Access and Control, ACM, N.Y., pp. 157-170.

PT6 MEHL, J. W., and WANG, C. P., "A Study of Order Transformation of Hierarchical Structures in IMS Data Bases," Proc. 1974 ACM-SIGFIDET Workshop on Data Description, Access and Control, ACM, N.Y., pp. 125-140.

PT7 HOUSEL, B. C., and HALSTEAD, M. H., "A Methodology for Machine Language Decompilation," Proc. of the 1974 ACM Annual Conference, ACM, N.Y., pp. 254-260.

PT8 HONEYWELL INFORMATION SYSTEMS, "Functional Specification Task 609 Data Base Interface Package," Defense Communications Agency Contract DCA 100-73-C-0055.

PT9 KINTZER, E., "Translating Data Base Procedures," Proc. 1975 ACM National Conference, ACM, N.Y., pp. 359-62.

PT10 SCHINDLER, S., "An Approach to Data Base Application Restructuring," Working Paper 76 ST 2.3, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Mich. 1976.

PT11 DALE, A. G., and DALE, N. B., "Schema and Occurrence Structure Transformation in Hierarchical Systems," Proc. 1976 International Conference on Management of Data, pp. 157-168.

PT12 SU, STANLEY Y. W., "Application Program Conversion Due to Data Base Changes," Proc. of the 2nd International Conference on VLDB, Brussels, Sept. 8-10, 1976, pp. 143-157.

PT13 SU, S. Y. W., and LIU, B. J., "A Methodology of Application Program Analysis and Conversion Based on Data Base Semantics," Proceedings of the International Conference on Management of Data,

1977, pp. 75-87.

PT14 HOUSEL, B. C., "A Unified Approach to Program and Data Conversion," Proceedings of the Third International Conference on Very Large Data Bases, ACM, N.Y., 1977, pp. 327-335.

PT15 SU, S. Y. W., and REYNOLDS, M. J., "Conversion of High-Level Sublanguage Queries to Account for Data Base Changes," Proc. of NCC, 1978, pp. 857-875.

(R) RESTRUCTURING

R1 FRY, J. P., and JERIS, D., "Towards a Formulation of Data Reorganization," Proc. 1974 ACM/SIGMOD Workshop on Data Description, Access and Control, ed. by R. Rustin, ACM, N.Y., pp. 83-100.

R2 SHU, N. C., HOUSEL, B. C., and LUM, V. Y., "CONVERT: A High-Level Translation Definition Language for Data Conversion," Comm. ACM 18,10, 1975, pp. 557-567.

R3 SHOSHANI, A., "A Logical-Level Approach to Data Base Conversion," Proc. 1975 ACM/SIGMOD International Conf. on Management of Data, ACM, N.Y., pp. 112-122.

R4 SHOSHANI, A., and BRANDON, K., "On the Implementation of a Logical Data Base Converter," Proc. International Conference on Very Large Data Bases, ACM, N.Y., 1975, pp. 529-531.

R5 HOUSEL, B. C., and SHU, N. C., "A High-Level Data Manipulation Language for Heirarchical Data Structures," Proc. of the 1976 Conference on Data Abstraction, Definition and Structure, Salt Lake City, Utah, pp. 155-169.

R6 NAVATHE, S. B., and FRY, J. P., "Restructuring for Large Data Bases: Three Levels of Abstraction," ACM Transactions on Data Base Systems, 1,2, ACM, N.Y., 1976, pp. 138-158.

R7 NAVATHE, S. B., "A Methodology for Generalized Data Base Restructuring," Ph.D. dissertation, The University of Michigan, 1976.

R8 GERRITSEN, ROB, and MORGAN, HOWARD, "Dynamic Restructuring of Data Bases With Generation Data Structures," Proc. of the 1976 ACM Conference, ACM, N.Y., pp. 281-286.

R9 SWARTWOUT, D., "An Access Path Specification Language for Restructuring Network Data Bases," Proc. of the 1977 SIGMOD Conference, ACM, N.Y., pp. 88-101.

R10 NAVATHE, S. B., "Schema Analysis for Data Base Restructuring," Proc. 3rd International Conference on Very Large Data Bases, ACM, N.Y., 1977. To appear in TODS.

R11 EDELMAN, J. A., JONES, E. E., LIAW, Y. S., NAZIF, Z. A., and SCHEIDT, D. L., "REORG - A Data Base Reorganizer," Bell Laboratories Internal Technical Report, April, 1976.

(SL) STORED-DATA DEFINITION

SL1 STORAGE STRUCTURE DEFINITION LANGUAGE TASK GROUP (SSDLTG) OF CODASYL SYSTEMS COMMITTEE, "Introduction to Storage Structure Definition" (by J. P. Fry); "Informal Definitions for the Development of a Storage Structure Definition Language" (by W. C. McGee); "A Procedural Approach to File Translation" (by J. W. Young, Jr.); "Preliminary Discussion of a General Data to Storage Structure Mapping Language" (by E. H. Sibley and R. W. Taylor), Proc. 1970 ACM-SIGFIDET Workshop on Data Description, Access and Control, ed. by E. F. Codd, Houston, Tex., Nov. 1970, pp. 368-80.

SL2 SMITH, D. C. P., "An Approach to Data Description and Conversion," Ph.D. dissertation, Moore School Report 72-20, University of Pennsylvania, Philadelphia, Pa., 1972.

SL3 TAYLOR, R. W., "Generalized Data Base Management System Data Structures and Their Mapping to Physical Storage," Ph.D. dissertation, The University of Michigan, Ann Arbor, Mich., 1971.

SL4 FRY, J. P., SMITH, D. C. P., and TAYLOR, R. W., "An Approach to Stored-Data Definition and Translation," Proc. 1972 ACM-SIGFIDET Workshop on Data Description, Access and Control, ed. by A. L. Dean, Denver, Colo., Nov. 1972, pp. 13-55.

SL5 BACHMAN, C. W., "The Evolution of Storage Structures," Comm. ACM 15,7 (July 1972), pp. 628-34.

SL6 SIBLEY, E. H. and TAYLOR, R. W., "A Data Definition and Mapping Language," Comm. ACM 16,12 (Dec. 1973), pp. 750-59.

SL7 HOUSEL, B., SMITH, D., SHU, N., and LUM, V., "Define: A Non-Procedural Data Description Language for Defining Information Easily," Proc. of 1975 ACM Pacific Conference, San Francisco, CA, April 1975, pp. 62-70.

SL8 The Stored-Data Definition and Translation Task Group, "Stored-Data Description and Data Translation: A Model and Language," Information Systems 2,3 (1977):95-148.

(SM) DATA SEMANTICS

SM1 SCHMID, H. A., and SWENSON, J. R., "On the Semantics of the Relational Model," Proc., ACM-SIGMOD 1975 Conference, May 1975, pp. 211-233.

SM2 ROUSSOPOULOS, N., and MYLOPOULOS, J., "Using Semantic Networks for Data Base Management," Proc. Very Large Data Base Conference, Framingham, Mass., Sept. 1975, pp. 144-172.

SM3 SU, STANLEY Y. W., and LO, D. H., "A Multi-level Semantic Data Model," CAASM Project, Technical Report No. 9, Electrical Engineering Dept., University of Florida, June 1976, pp. 1-29.

(TA) TRANSLATION APPLICATIONS

TA1 WINTERS, E. W., and DICKEY, A. F., "A Business Application of Data Translation," Proceedings of the 1976 SIGMOD International Conference on Management of Data, Ed. by J. B. Rothnie, Washington, D.C., June 1977, pp. 189-196.

(UR) UM RESTRUCTURING

UR1 LEWIS, K., DRIVER, B., and DEPPE, M., "A Translation Definition Language for the Version II Translator," Working Paper 809, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

UR2 LEWIS, K., and FRY, J., "A Comparison of Three Translation Definition Languages," Working Paper DT 5.1, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

UR3 DEPPE, M. E., "A Relational Interface Model for Data Base Restructuring," Technical Report 76 DT 3, Data

Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UR4 DEPPE, M. E., LEWIS, K. H., and SWARTWOUT, D. E., "Restructuring Network Data Bases: An Overview," Data Translation Project, Technical Report 76 DT 5, The University of Michigan, Ann Arbor, Michigan, 1976.

UR5 DEPPE, M. E., and LEWIS, K. H., "Data Translation Definition Language Reference Manual for Version IIA Release 1," Data Translation Project, Working Paper 76 DT 5.2, The University of Michigan, Ann Arbor, Michigan, 1976.

UR6 SWARTWOUT, D. E., MARINE, A. M., and BAKKOM, D. E., "Partial Restructuring Approach to Data Translation," Data Translation Project, Working Paper 76 DT 8.1, The University of Michigan, Ann Arbor, Michigan, 1976.

UR7 SWARTWOUT, D. E., WOLFE, G. J., and BURPEE, C. E., "Translation Definition Language Reference Manual for Version IIA Translator, Release 3," Data Translation Project, Working Paper 77 DT 5.3, The University of Michigan, Ann Arbor, Michigan, 1977.

(US) UM STORED-DATA DEFINITION

US1 DATA TRANSLATION PROJECT, "Stored-Data Definition Language Reference Manual," The University of Michigan, Ann Arbor, Michigan, 1972.

US2 DATA TRANSLATION PROJECT, "Revised Stored-Data Definition Language Reference Manual," The University of Michigan, Ann Arbor, Michigan, 1974.

US3 DATA TRANSLATION PROJECT, "University of Michigan Stored-Data Definition Language Reference Manual for Version II Translator," The University of Michigan, Ann Arbor, Michigan, 1975.

US4 BIRSS, E. W., and FRY, J. P., "A Comparison of Two Languages for Describing Stored Data," Data Translation Project, Technical Report 76 DT1, The University of Michigan, Ann Arbor, Michigan, 1976.

(UT) UM TRANSLATION

UT1 "Functional Design Requirements for a Prototype Data Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1972.

UT2 "Design Specifications of a Prototype Data Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1972.

UT3 "Program Logic Manual for the University of Michigan Prototype Data Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1973.

UT4 "Users Manuals for the University of Michigan Prototype Data Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1973.

UT5 "Functional Design Requirements of the Version I Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1973.

UT6 "Program Logic Manual for the University of Michigan Version I Data Translator," Working Paper 306, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1974.

UT7 "Design Specifications: Version II Data Translator," Working Paper 307, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

UT8 BIRSS, E., DEPPE, M., and FRY, J., "Research and Data Reorganization Capabilities for the Version IIA Data Translator," Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1975.

UT9 BIRSS, E., et al., "Program Logic Manual for the Version IIA Data Translator," Working Paper 76 DT 3.1, Data Translation Projects, The University of Michigan, Ann Arbor, Michigan, 1976.

UT10 BODWIN, J., et al., "Data Translator Version IIA Release 1 User Manual," Working Paper 76 DT 3.2, Data Translation Project, The University of Michigan, Ann Arbor, Michigan, 1976.

UT11 BODWIN, J., et al., "Data Translator Version IIA Release 2 User Manual," Working Paper 76 DT 3.4, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1976.

UT12 KINTZER, E., et al., "Michigan Data Translator Version

IIB Release 1 User Manual," Technical Paper 77 DT 8, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.

UT13 BURPEE, C. E., et al., "Michigan Translator Program Logic Manual Version IIB Release 1," Working Paper 77 DT 3.7, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.

UT14 BAKKOM, D., et al., "Specifications for a Generalized Reader and Interchange Form," Working Paper 77 DT 6.2, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.

UT15 DeSMITH, D., and HUTCHINS, L., "Michigan Data Translator Design Specifications Version IIB," Working Paper 77 DT 3.8, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.

UT16 KINTZER, E., et al., "Michigan Data Translator Version IIB Release 1.1 User Manual," Technical Paper 77 DT 8.1, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Michigan, 1977.

UT17 BAKKOM, D. E., and Schindler, S. J., "Operational Capabilities for Data Base Conversion and Restructuring," Technical Report 77 DT 6, Data Base Systems Research Group, The University of Michigan, Ann Arbor, Mich., 1977.

(Z) RELATIONAL SYSTEM

Z5 CHAMBERLAIN, D. C. and BOYCE, R. F., "SEQUEL: A Structured English Query Language," Proceedings of the ACM-SIGMOD Workshop on Data Description, Access and Control, ACM, N.Y., 1974.

7. PARTICIPANTS

The following is a list of attendees, participants and contributors to the workshop.

Edward Arvel
Conversion Experiences
Data Sciences Group
890 National Press Building
Washington, D.C. 20045

Marty Aronoff
Management Objectives
National Bureau of Standards
Tech B258
Washington, D.C. 20234

Robert Bemer
Standards
Honeywell Information Systems
P.O. Box 6000
Phoenix, AZ 85005

John Berg
Proceedings Editor
National Bureau of Standards
Tech A259
Washington, D.C. 20234

Edward Birss
Conversion Technology
Hewlett Packard
General Systems Division
5303 Stevens Creek Blvd.
Bldg. 498-3
Santa Clara, CA 95050

Don Branch
Standards
Advisory Bureau for Computing
Room 828, Lord Elgin Plaza
66 Slatter Street
Ottawa, Ontario K1A 0T5
CANADA

Jean Bryce
Standards
M. Bryce & Associates, Inc.
1248 Springfield Pike
Cincinnati, OH 45215

Milt Bryce Chairman, Standards	M. Bryce & Associates, Inc. 1248 Springfield Pike Cincinnati, OH 45215
Jim Burrows Chairman, Conversion Experiences	Director, Institute for Computer Sciences and Technology National Bureau of Standards Administration Bldg., Room A200 Washington, D. C. 20234
Richard G. Canning Management Objectives	Canning Publications, Inc. 925 Anza Avenue Vista, CA 92083
Lt. Michael Carter Conversion Experiences	Air Force Data Systems Design Ctr. AFDSDC/SDDA, Building 857 Gunter AFB, AL 36114
Joseph Collica Conversion Experiences	National Bureau of Standards Tech. A254 Washington, D. C. 20234
Elizabeth Courte Conversion Experiences	Bell Laboratories 3B210 Six Corporate Plaza Piscataway, NJ 08854
Ahron Davidi Conversion Experiences	Blue Cross of Massachusetts 100 Summer Street, 12th Floor Boston, Massachusetts 02106
Peter Dressen Conversion Technology	Honeywell Inforamtion Systems P. O. Box 6000 Phoenix, Arizona 85005
Ruth F. Dyke Conversion Experiences	U.S. Civil Service Commission 1900 E Street, N. W., Room 6410 Washington, D. C. 20415
Larry Espe Management Objectives	Nolan, Norton and Company One Forbes Road Lexington, Massachusetts 02173

Gordon Everest Management Objectives	University of Minnesota 271 19 Avenue South Minneapolis, MN 55455
Elizabeth Fong Standards	National Bureau of Standards Tech B212 Washington, D.C. 20234
James P. Fry Chairman, Conversion Technology	276 Business Administration University of Michigan Ann Arbor, MI 48109
Al Gaboriault Standards	Sperry Univac P.O. Box 500 Mail Station C1NW-12 Blue Bell, PA 19424
Mr. Rob Gerritsen Management Objectives	Wharton School University of Pennsylvania Philadelphia, Pennsylvania 19174
Richard Godlove Management Objectives	Monsanto Company 800 North Lindbergh Boulevard St. Louis, Missouri 63166
Nancy Goguen Conversion Technology	Bell Laboratories 6 Corporate Place Piscataway, NJ 08854
Seymour Jeffery Host	Director, Center for Programming Science and Technology National Bureau of Standards Tech A247 Washington, D.C. 20234
Samuel C. Kahn Management Objectives	Information System Dep. - Planning E. I. duPont de Nemours & Co. Wilmington, Delaware 19899
Mike Kaplan Conversion Technology	Bell Laboratories 8 Corporate Place Piscataway, NJ 08854

Anthony Klug Standards	Computer Sciences Department University of Wisconsin Madison, Wisconsin 53706
Henry Lefkovits Standards	H. C. Lefkovits & Associates, Inc. P.O. Box 297 Harvard, MA 01451
H. Eugene Lockhart Management Objectives	Nolan, Norton & Company One Forbes Road Lexington, Massachusetts 02173
Thomas Lowe Host	Chief Operations Engineering Division Center for Programming Science and Technology National Bureau of Standards Tech A265 Washington, D.C. 20234
Gene Lowenthal Conversion Technology	MRI Systems Corporation P.O. Box 9968 Austin, Texas 78766
Vincent Lum Conversion Technology	IBM Research Corp., K55/282 5600 Cottle Road San Jose, CA 95103
Mr. John Lyon Management Objectives	Colonial Penn Group Data Corporatio 5 Penn Center Plaza Philadelphia, PA 19103
Halaine Maccabee Conversion Experiences	Northeastern Illinois Plan. Comm. (N 400 West Madison St. Chicago, Illinois 60606
William Madison Contributor	Universal Systems, Inc. 2341 Jefferson Davis Highway Arlington, Virginia 22202

Daniel B. Magraw General Chairman	State of Minnesota Department of Administration 208 State Administration Building St. Paul, MN 55155
Robert Marion Conversion Technology	Defense Communications Agency 11440 Issac Newton Square, North Reston, VA 22090
Steven Merritt Conversion Experiences	GAO, FGMS-ADP, Room 6011 411 G Street, N.W. Washington, D.C. 20548
Jack Minker ACM Liaison	Chairman Department of Computer Science University of Maryland College Park, Maryland 20742
Thomas E. Murray Management Objectives	Delmonte Corp. P.O. Box 3575 San Francisco, CA 94119
Shamkant Navathe Conversion Technology	New York University Grad. School of Business 600 Tisch Hall 40 West 4th Street New York, New York 10003
Mr. Jack Newcomb Management Objectives	Department of Finance & Admin. 326 Andrew Jackson State Off. Bldg. Nashville, TN 37219
Richard Nolan Chairman, Management Objectives	Nolan, Norton and Company One Forbes Road Lexington, Massachusetts 02173
T. William Olle Management Objectives	Consultant 27 Blackwood Close West Byfleet Surrey, KT14 6PP ENGLAND
Mayford Roark Keynoter	Ford Motor Company The American Road Dearborn, MI 48121

C. H. Rutledge Standards	Marathon Oil Company 539 South Main Street Findley, OH 45840
Mr. Michael J. Samek Management Objectives	Celanese Corporation 1211 Avenue of the Americas New York, NY 10036
Steve Schindler Management Objectives and Conversion Technology	University of Michigan 276 Business Administration Ann Arbor, MI 48109
Mr. Richard D. Secrest Management Objectives	Standard Oil Company (Indiana) P.O. Box 591 Tulsa, OK 74102
Philip Shaw Standards	IBM Corporation 555 Bailey Avenue San Jose, CA 95150
Arie Shoshani Conversion Technology	Lawrence Berkeley Laboratory University of California Berkeley, CA 94720
Edgar Sibley Management Objectives	Dept. of Info. Systems Mgmt. University of Maryland College Park, MD 20742
Prof. Diane Smith Conversion Technology	University of Utah Merrill Engineering Bldg. Salt Lake City, UT 84112
Alfred Sorkowitz Conversion Experiences	Department of Housing & Urban Devel. 451 7 Street, S.W., Room 4152 Washington, D.C. 20410
Prof. Stanley Su Conversion Technology	Electrical Engineering University of Florida Gainesville, FL 32611

Donald Swartwout Conversion Technology	276 Business Administration University of Michigan Ann Arbor, MI 48109
Robert W. Taylor Conversion Technology	IBM Corporation IBM Research, K55/282 5600 Cottle Road San Jose, CA 95193
Jay Thomas Standards	Allied Van Lines 2501 West Roosevelt Road Broadview, IL 60153
Ewart Willey Standards	Prudential Assurance Co. 142 Holborn Bars London EC1N 2NH ENGLAND
Major Jerry Winkler Standards	U.S.A.F. -AFDSC/GKD Room 3A153, Pentagon Washington, D.C. 20030
Beatrice Yormark Conversion Technology	Interactive Systems Corporation 1526 Cloverfield Blvd. Santa Monica, CA 90404

A Contributor is one who could not attend but either submitted a paper or added to the Workshop in a manner that deserves our acknowledgement and thanks.

*U S GOVERNMENT PRINTING OFFICE: 1980 328-117/6522

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-64	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

DATA BASE DIRECTIONS - THE CONVERSION PROBLEM
Proceedings of a Workhop held in Ft. Lauderdale, FL, Nov. 1-3, 1977

5. Publication Date: September 1980

6. Performing Organization Code

7. AUTHOR(S)

John L. Berg, Editor

8. Performing Organ. Report No.

9. PERFORMING ORGANIZATION NAME AND ADDRESS

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, DC 20234

10. Project/Task/Work Unit No.

11. Contract/Grant No.

12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)*

National Bureau of Standards
Department of Commerce
Washington, DC 20234

Assoc. for Computing Machinery
1133 Ave. of Americas
New York, NY 10036

13. Type of Report & Period Cover: Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

What information can help a manager assess the impact a conversion will have on a data base system, and of what aid will a data base system be during a conversion? At a workshop on the data base conversion problem held in November 1977 under the sponsorship of the National Bureau of Standards and the Association for Computing Machinery, approximately seventy-five participants provided the decision makers with useful data.

Patterned after the earlier Data Base Directions Workshop, this workshop, Data Base Directions - the Conversion Problem, explores data base conversion from four perspectives: management, previous experience, standards, and system technology. Each perspective was covered by a workshop panel that produced a report included her

The management panel gave specific direction on such topics as planning for dat base conversions, impacts on the EDP organization and applications, and minimizing the impact of the present and future conversions. The conversion experience panel drew upon ten conversion experiences to compile their report and prepared specific checklists of "do's and don'ts" for managers. The standards panel provided comments on standards needed to support or facilitate conversions and the system technology panel reports comprehensively on the systems and tools needed with strong recommendations on future research.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Conversion; data base; data-description; data-dictionary; data-directory; data-manipulation; DBMS: languages; query

18. AVAILABILITY

☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☒ Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PA
UNCLASSIFIED	178
20. SECURITY CLASS (THIS PAGE)	22. Price
UNCLASSIFIED	$5.50

USCOMM-DC

CHECK THEM OUT

do those automated checkout counters, gas pumps, it offices, and banking facilities work? What every sumer should know about the modern electronic sys- now used in everyday transactions is explained in -page booklet published by the Commerce Depart- t's National Bureau of Standards.

mation in the Marketplace (NBS Consumer Informa- Series No. 10) is for sale by the Superintendent of uments, U.S. Government Printing Office, Washington, 20402. Price: 90 cents. Stock No. 003-003-01969-1.

"AUTOMATION IN THE MARKETPLACE"
A Consumer's Guide

D/H CAKE MIX .83
WALNUTS CAN .59
JELLO PUDDING .30

▶ UNIVERSAL PRODUCT CODE see booklet

1 GREEN PEPPER .34
LASER SCANNER see bklt
CHRRY TOMATO .79

▶ ELECTRONIC CASH REGISTER see bklt

1 CUCUMBERS .34

▶ HANDLING OF UNCODED ITEMS see bklt

▶ ELECTRONIC SCALES see bklt

GREETING CARD .60

▶ WEIGHTS & MEASURES ENFORCEM'T see bklt

DELICATESSEN 1.35
2.19b @49/bBROCCO 1.07

▶ SPECIAL FEATURES OF COMPUTER CHECKOUT SYSTEMS see bklt

DRUG 4.49 T

▶ BANK TELLER MACHINES see bklt

▶ COMPUTER TERMINALS see bklt

▶ CONSUMER ISSUES see bklt

▶ THANK YOU BE INFORMED see bklt

(please detach along dotted line)

RDER FORM

ASE SEND ME ___ COPIES OF

omation in the Marketplace

0 per copy.

Stock No. 003-003-01969-1

I enclose $_____ (check, or money order) or charge my Deposit Account No._____.

Total amount $_____.

Make check or money order payable to Superintendent of Documents.

MAIL ORDER FORM WITH PAYMENT TO

Superintendent of Documents
U.S. Government Printing Office
Washington, D.C. 20402

or any U.S. Department of Commerce district office

e type or print)

E_____

ESS_____

_____

E_____ ZIP CODE_____

FOR USE OF SUPT. DOCS.

___Enclosed _____
To be mailed
___later _____
Refund _____
Coupon refund _____
Postage _____

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent Bureau publications in both NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $13; foreign $16.25. Single copy, $3 domestic; $3.75 foreign.

NOTE: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS—This monthly magazine is published to inform scientists, engineers, business and industry leaders, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing. Annual subscription: domestic $11; foreign $13.75.

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a worldwide program coordinated by NBS under the authority of the National Standard Data Act (Public Law 90-396).

NOTE: The principal publication outlet for the foregoing data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St., NW, Washington, DC 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The standards establish nationally recognized requirements for products, and provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order the **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, DC 20402.*

Order the **following** *NBS publications—FIPS and NBSIR's—from the National Technical Information Services, Springfield, VA 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. The Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services, Springfield, VA 22161, in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service. A literature survey issued biweekly. Annual subscription: domestic $35; foreign $45.

Liquefied Natural Gas. A literature survey issued quarterly. Annual subscription: $30.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $45. Please send subscription orders and remittances for the preceding bibliographic services to the National Bureau of Standards, Cryogenic Data Center (736) Boulder, CO 80303.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

SPECIAL FOURTH-CLASS RATE
BOOK