ENCE & TECHNOLOGY:

DATA BASE REORGANIZATION — PRINCIPLES AND PRACTICE

NBS Special Publication 500-47

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress 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 users in the public and private sectors to address national needs and to solve national problems in the public interest; conducts research in engineering and applied science in support of objectives in 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 utlimate 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 divisions:

Systems and Software — Computer Systems Engineering — Information Technology.

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

The National Bureau of Standards was reorganized, effective April 9, 1978.

COMPUTER SCIENCE & TECHNOLOGY:
Data Base Reorganization — Principles and Practice

Gary H. Sockut

Application Systems Division
Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

and

Robert P. Goldberg

BGS Systems, Inc.
Box 128
Lincoln, MA 01773

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary

Jordan J. Baruch, Assistant Secretary for Science and Technology

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued April 1979

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 publication.

National Bureau of Standards Special Publication 500-47

Nat. Bur. Stand. (U.S.), Spec. Publ. 500-47, 51 pages (Apr. 1979)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 79-600055

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1979

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Stock No. 003-003-02055-9 Price $2.30
(Add 25 percent additional for other than U.S. mailing).

CONTENTS

FIGURES

DATA BASE REORGANIZATION -- PRINCIPLES AND PRACTICE

Gary H. Sockut
and
Robert P. Goldberg

Data base reorganization can be defined as changing some aspect of the way in which a data base is arranged logically and/or physically. This paper contains tutorials and surveys. It introduces the basic concepts of reorganization, including why it is performed. Many examples of types of reorganization are described and are classified into logical / physical levels. The paper then covers pragmatic issues such as reorganization strategies, a survey of several commercial reorganization facilities, case studies, and data base administration considerations. Finally, several research efforts are surveyed.

Key words: data base; data base management; reorganization; restructuring; file maintenance.

INTRODUCTION

We define data base reorganization as changing some aspect of the way in which a data base is arranged. We use reorganization as a generic term that covers what some authors call restructuring (changing logical structures) and reformatting (changing physical structures). Some examples of reorganization are adding an attribute, changing a relationship between one to one and one to many, deleting a secondary index, changing between sequential and pointer linkages, changing between hashed and indexed access, and eliminating overflow in an indexed sequential access method.

The intended audience for this paper includes data processing operations personnel, data base researchers, and students. The paper should be comprehensible to a reader with an introductory knowledge of data bases. The reader should be acquainted with terms such as data base, data base management system (DBMS), schema, subschema, record, CODASYL (or DBTG) set, and access method. Such introductory

knowledge can be acquired from sources such as SIBL76, DATE77, or MART77.

Reorganization may be performed for a variety of reasons. Reorganization may be highly desirable (e.g. to improve performance, storage utilization, or human productivity), or reorganization may be necessary (e.g. to change security policies, to create a new data base from old data bases, or to improve functional capabilities). Below are some examples of circumstances under which reorganization is appropriate:

- The definition of information changes. For example, if a company initially requires each of its employees to work on only one project at any time, but later the company's policy is changed to allow an employee to work on several projects simultaneously, then the one to many relationship between projects and employees must be changed to a many to many relationship.

- A new type of information is added to a data base. This may require increasing the size of a record type to accommodate a new field.

- New legislation requires a change. For example, restricting disclosure of information among government agencies may require splitting records into disclosable and non-disclosable parts.

- A new data base is created from old data bases or files. For example, a company that acquires another company may decide to merge the two customer data bases, which may be associated with different DBMS and which may be in different formats, thus requiring conversion.

- Empirical characteristics of information change. For example, in a file of families, the optimal amount of space to reserve for children near a family record is the result of a time - space trade-off. As zero population growth becomes more popular, more families are small, and the optimal amount of space to reserve may decrease.

- Characteristics of usage change, on either a short-term or a long-term basis. For example, if new sociological research using a population data base requires access via a particular key, a new secondary index might be added.

- As the amount of information grows, a data base may be moved to larger or faster storage devices. This may require modifying the mapping of records to physical locations.

- Observation of data base performance may lead to "tuning" or redesigning aspects such as hashing parameters.

- The optimal access arrangement varies with time, as in an airline reservation system, where flight information for the next few days may be finely indexed for quick access while information for later days may be coarsely indexed to save space. At the end of the day, today's information is archived, and one more day's information is indexed finely.

- Performance can degrade during normal operation as unpredictable insertions are made or deleted records accumulate. For example, if insertions into an indexed sequential file are clustered in one key range, access times for that part of the file increase as overflow structures accumulate. Maintenance may produce a more balanced structure yielding better access times.

Some storage structures may be selected and maintained automatically by a DBMS. Even for such a system, those storage structures, as well as logical structures, may require manual or automatic reorganization at times.

The rest of this paper contains tutorials and surveys. Many examples of types of reorganization are described and are classified into logical / physical levels in Section 1. This is followed by a discussion of pragmatic issues in Section 2. These include (1) strategies used to reorganize, (2) a survey of reorganization facilities provided with several well-known commercial DBMS, (3) case studies, and (4) data base administration considerations. We then survey several research efforts in this area in Section 3.

At several places in this paper, we describe selected commercial systems. The inclusion or exclusion of a system, as well as the order and content of our descriptions, do not imply endorsement or disapproval by the authors or their organizations. We do not certify that the systems operate as described.

1. TYPES OF REORGANIZATION

This section describes many examples of types of reorganization, which are classified using a data base accessing model. As explained below, the accessing model describes data base constructs at each of several levels ranging from logical to physical. Types of reorganization are classified into levels according to the constructs they change. Reorganization can be performed at any level. The classification is presented primarily for pedagogic purposes, but it can also be used to illustrate data independence; i.e. constructs at high levels are unaffected by changes in constructs at low levels, except for differences in performance.

1.1 Overview of the Classification

The classification uses the stratification of DIAM II [SENK75], the second version of the Data Independent Accessing Model, which was developed by Michael E. Senko. DIAM II consists of a data model and an accessing model. A data model is a collection of data structures, occurrences of which represent all logical information in a data base. Examples are the CODASYL (or DBTG) [CODA71, CODA78], relational [CODD70], IMS [IBM77a], and entity-relationship [CHEN75, CHEN78] data models, as well as the many data models described in KERS76. An accessing model is a definition of how data is physically stored and accessed.

The classification is useful for all data models. It uses only DIAM II's accessing model, not its data model. Most reorganization examples used below are taken from CODASYL and IMS implementations, and some are taken from the relational model. DIAM I [ASTR72], an earlier version of DIAM, has also been used to describe relational implementations [SCHN76].

We selected DIAM because it provides a stratification of data base constructs and because it is well-known. The stratification used by the CODASYL Stored-Data Definition and Translation Task Group [CODA77] was also largely based upon DIAM's stratification. The classification appears to be general enough to apply to a large variety of existing data base structures, but if necessary, other classifications could be developed for other structures.

The levels of DIAM II are described briefly below. This is not an in-depth description, but it provides sufficient criteria for classifying types of reorganization. The references contain more detailed explanations of the DIAM II levels.

Figure 1 shows an overview of DIAM II. Logical levels appear at the top, and physical levels appear at the bottom. Rectangles represent constructs at the various levels, while ovals represent inter-level mappings between constructs. The five DIAM II levels are labelled at the left, while the three levels of the ANSI SPARC data base architecture [TSIC77] (a proposed architectural framework for data base systems) are labelled at the right. Senko has pointed out a correspondence [SENK75] between ANSI SPARC and DIAM II levels: The ANSI SPARC external schema corresponds to the DIAM II end-user level, the conceptual schema corresponds to the infological level, and the internal schema corresponds to the string level, encoding level, and physical device level. Some data base terms in common use (subschema, schema, and access methods) that correspond (at least roughly) to DIAM II concepts also appear in the figure.

We begin by describing the infological level [SENK75], which defines attributes and relationships among them. Attributes' logical representations (e.g. range of values) are also defined here. This level corresponds roughly to the common notion of schema, although schemas in existing DBMS generally contain information from lower DIAM II levels as well.

The end-user level [SENK75] provides users (or applications) with views of the infological level's constructs. A user may view a subset of the data base's attributes and relationships, and this subset may be viewed as having a certain structure (e.g. a hierarchy). This level corresponds roughly to the common notion of subschema.

The String Level [SENK75] defines access paths, e.g. to implement relationships. Strings are linkages among attributes (e.g. to define a CODASYL record type in terms of fields) or among strings (e.g. to define a CODASYL set in terms of an owner record type and a member record type). Constructs (attributes or strings) that are linked by a string can be ordered within the string. There may be a choice of several possible string representations for a given relationship. For example, a one to many relationship might be implemented in CODASYL as a set or as a repeating group of fields. Strings are also used in the implementation of secondary indices.

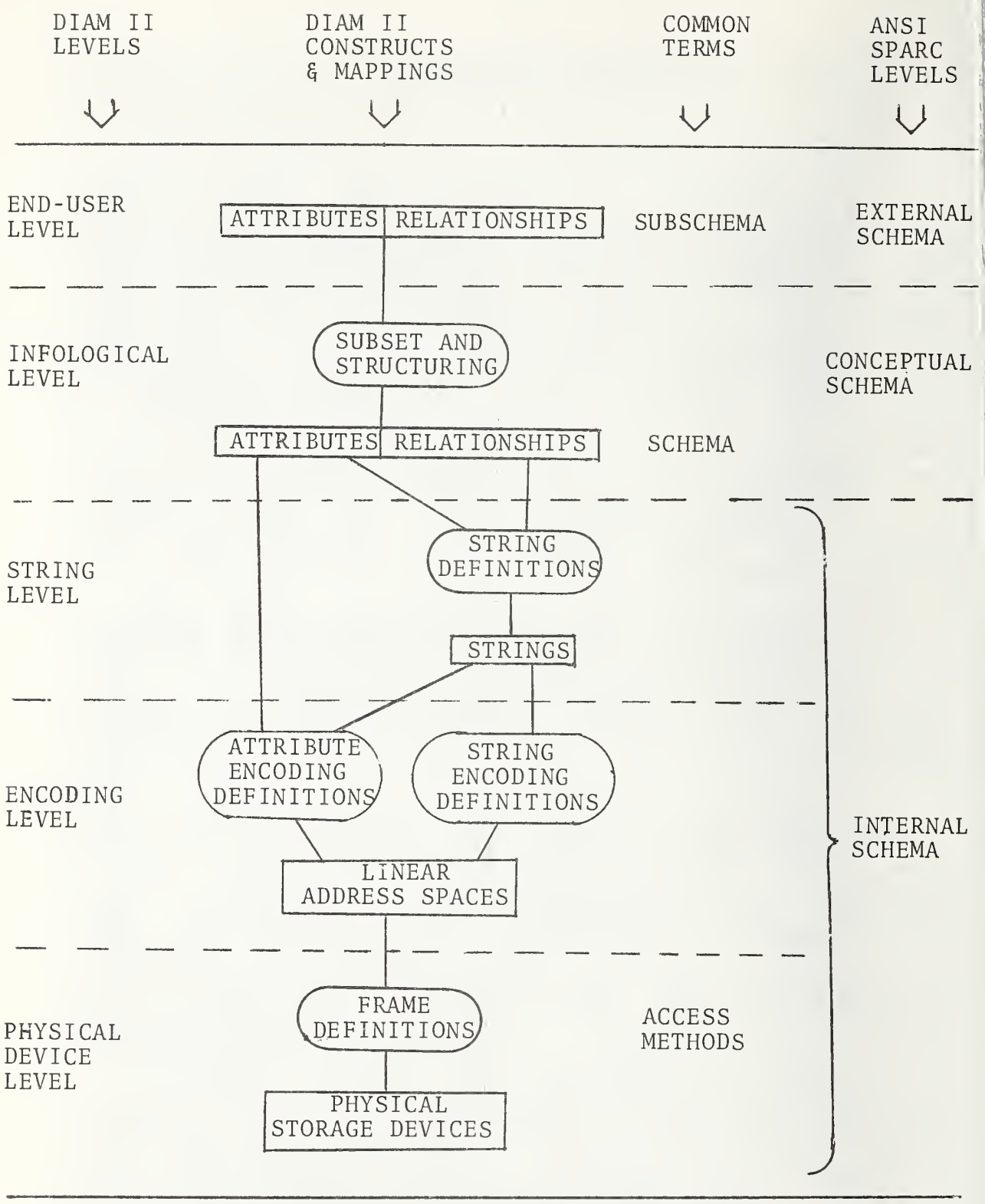


FIGURE 1

OVERVIEW OF DIAM II

The encoding level [ALTM72] defines how strings and attributes are physically represented (encoded) as bits in one-dimensional bit streams (called linear address spaces). Strings can be represented by physical contiguity (as in a record) or by pointers (as in a set). Examples of attribute encoding are character codes and binary or decimal integers.

The physical device level [SENK76] maps linear address spaces onto physical storage devices, using constructs called frames as an intermediate stage. A frame is a generalization of physical units such as block, track, cylinder, page, and disk pack. Frame definitions correspond roughly to access methods.

Sections 1.2 - 1.5 classify many examples of types of reorganization within the levels of DIAM II. Subdivisions within a level are used for pedagogic purposes, not for finer stratification. Most examples were taken from CODASYL and IMS implementations and from the relational data model, because these systems are well-known. CODASYL, IMS, and relational terms and constructs used are described in depth in the references (e.g. CODA78 for CODASYL, IBM77a for IMS, and CODD70 for relational). A type of reorganization in an existing DBMS may correspond to more than one DIAM level.

1.2 Reorganization at the Infological Level

Definitions of attributes and relationships are changed in the examples shown in this section. The discussion applies to any data model. A similar collection of changes is described in CHEN77 for the entity-relationship data model.

In cases in which infological level changes are hidden from application programs via unchanged subschemas, old application programs are not changed. If a subschema is changed, the change affects application programs that use the subschema. Application program conversion, which is not covered in this paper, can be more costly than data conversion [HOUS77]. We do not address end-user level changes separately from infological level changes. Infological level changes are described below:

Attributes can be added, deleted, combined, split, or renamed. For example, in a relational data base, a relation's degree (number of attributes) may be changed.

Attributes' logical representations can be changed. This may affect fields' and records' sizes, which are specified at the encoding level. These changes include scale (e.g. inches vs. cm.; monthly vs. semi-monthly salary) and range of values (including logical field size). Alternatively, some such changes might be performed at the encoding level, if it is desired to insulate the infological level from the changes.

Security controls can be changed. Such changes might be implemented at lower levels.

Relationships can be changed, although such changes are implemented at the string level, as described in Section 1.3.1. These changes include:

- (1) Creating, destroying, or renaming a relationship.
- (2) Changing among a one to one, a one to many, and a many to many relationship.
- (3) In a one to many relationship, moving an attribute between the one and the many.

1.3 Reorganization at the String Level

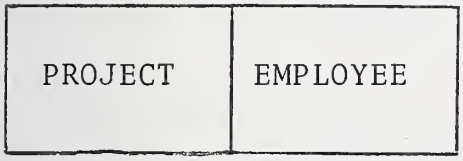
This section's examples involve changes in string definitions. The examples are divided into (1) implementing changes in relationships, (2) re-implementing unchanged relationships, and (3) other changes at the string level.

1.3.1 Implementing Changes in Relationships

The three relationship changes listed at the end of Section 1.2 can be implemented, respectively, by the string level operations described below:

(1) Create, destroy, or rename a string (e.g. CODASYL record, IMS segment, CODASYL set, or IMS hierarchy).

(2) Change among two groups of fields in a single record type, two record types related as parent and child, and three record types related as parent, relator, and parent. These changes are illustrated for the CODASYL data model using three data structure diagrams [BACH69] in Figure 2. The diagrams represent the changing relationship between a

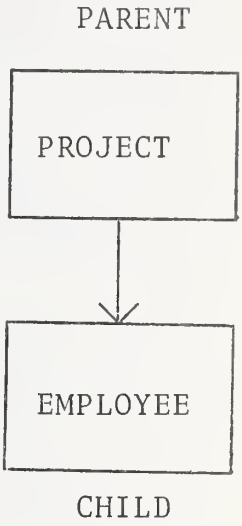


(1) ONE TO ONE RELATIONSHIP.

TWO GROUPS OF FIELDS IN A SINGLE RECORD TYPE.

(2) ONE TO MANY RELATIONSHIP.

TWO RECORD TYPES RELATED AS PARENT AND CHILD.



(3) MANY TO MANY RELATIONSHIP.

THREE RECORD TYPES RELATED AS PARENT, RELATOR, AND PARENT.

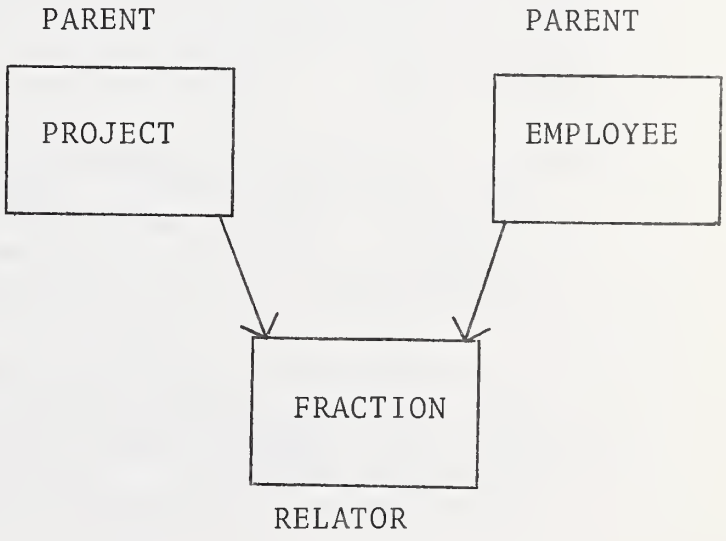


FIGURE 2

DATA STRUCTURE DIAGRAMS TO ILLUSTRATE IMPLEMENTING ONE TO ONE, ONE TO MANY, AND MANY TO MANY RELATIONSHIPS

company's projects and the company's employees. Similar changes could be illustrated in the relational data model by using one, two, and three relations, respectively.

- In the first diagram, each project employs exactly one employee, who works on only that one project. There is a one to one relationship between projects and employees, represented by two groups of fields in a single record type.

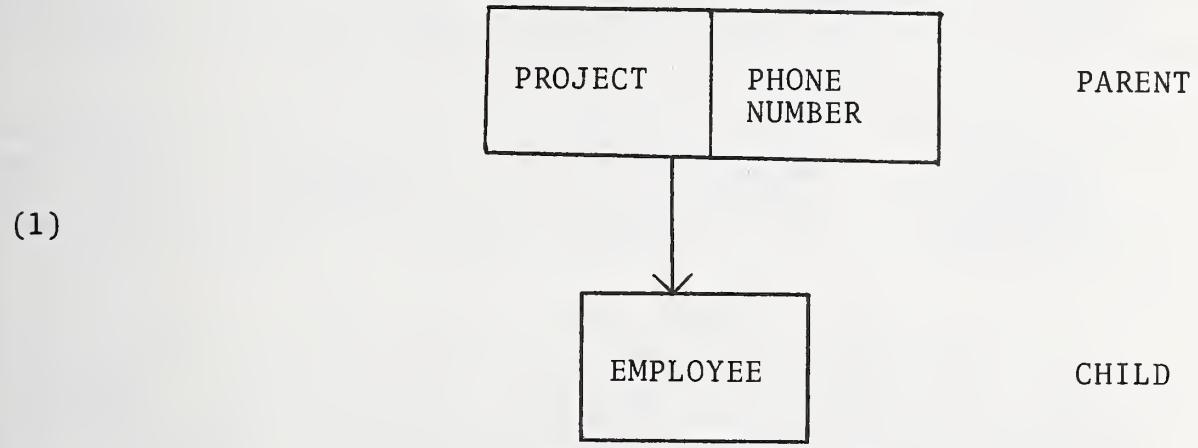
- In the second diagram, each project employs any number of employees, each of whom works on only that one project. There is a one to many relationship between projects and employees, represented by two record types related as parent and child.

- In the third diagram, each project employs any number of employees, each of whom works on any number of projects. There is a many to many relationship between projects and employees, represented by three record types related as parent, relator, and parent, where the relator record indicates the fraction of a given employee's time that is spent on a given project.

(3) Move a field between parent and child, as shown using two data structure diagrams in Figure 3. The diagrams represent the relationships among projects, employees, and phone numbers. In both diagrams, there is a one to many relationship between projects and employees, represented by project in a parent record type and employee in a child record type.

- In the first diagram, all the employees on a project have the same phone number, since the projects operate on low budgets. The attribute "PHONE NUMBER" is associated with a project and is represented as a field in the parent record type. In the relational model, it would be an attribute in a project relation.

- In the second diagram, the projects are well funded, and each employee may have his or her own phone number. The attribute "PHONE NUMBER" is represented as a field in the child record type. In the relational model, it would be an attribute in an employee relation.



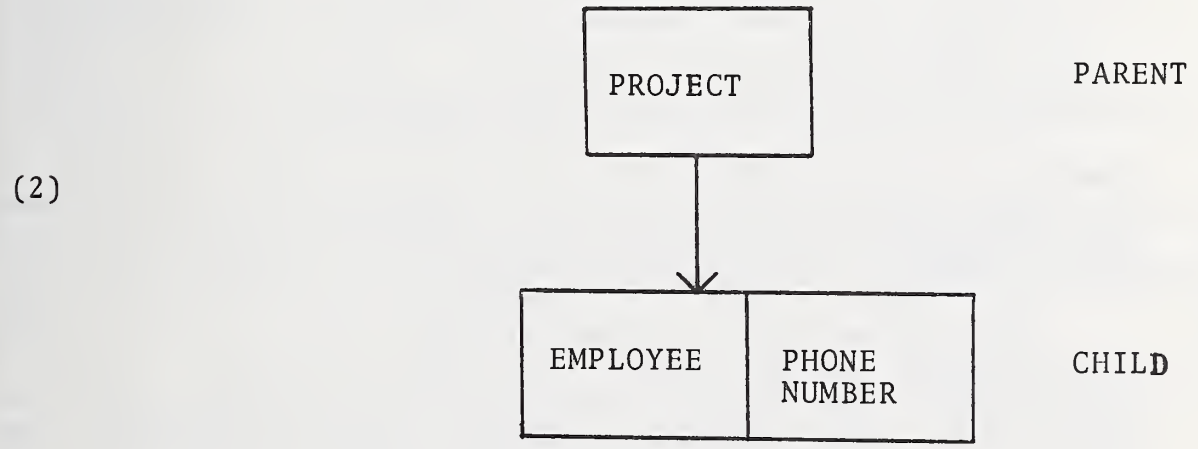


FIGURE 3

DATA STRUCTURE DIAGRAMS TO ILLUSTRATE
MOVING A FIELD BETWEEN PARENT AND CHILD

1.3.2 Re-implementing Unchanged Relationships

Unchanged relationships can be re-implemented with different strings. Three types of re-implementation are shown below:

(1) Change the implementation of a one to many relationship. For example, Figure 4 shows diagrams of record occurrences (not data structure diagrams) for three possible CODASYL implementations of a one to many relationship between projects and employees. A tutorial writing project employs Sockut and Goldberg, while a declaration writing project employs Jefferson.

- In the first diagram, there is a single record type which contains a non-repeating field (or group of fields) for project and a repeating field (or group of fields) for employee. There is one record occurrence for each project.

- In the second diagram, there are two record types, where the parent contains a non-repeating field for project and the child contains a non-repeating field for employee. There is one parent record occurrence for each project and one child record occurrence for each employee. The University of Michigan's Data Translation Project calls this arrangement expanded [NAVA75].

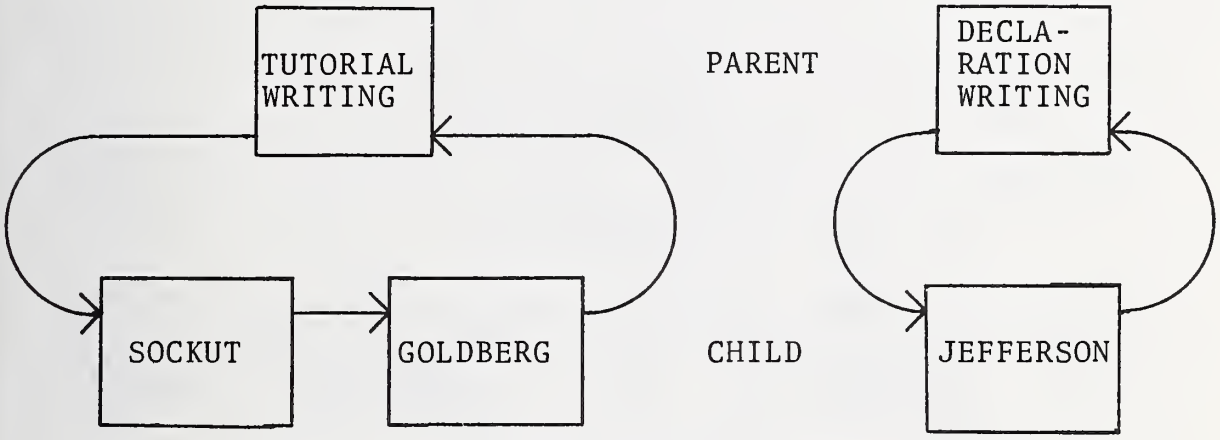
- In the third diagram, there is a single record type which contains a non-repeating field for project and a non-repeating field for employee. There is one record occurrence for each employee. The Data Translation Project calls this arrangement compressed.

Changing between embedded attribute values and pointers to attribute values is similar to changing between compressed and expanded, respectively, where an OWNER pointer is used in the expanded case to point to the attribute value.

(2) Change the implementation of a one to many relationship where there are two (or more) types of descendants. For example, Figure 5 shows data structure diagrams for three possible CODASYL implementations of a one to many relationship between projects and employees where we wish to distinguish permanent from temporary employees. In all three diagrams, there is one parent record type.

- In the first diagram, there is one child record type, which contains a field (LONGEVITY) that indicates permanent or temporary. The Data Translation Project and the entity-relationship model call this arrangement

(1)



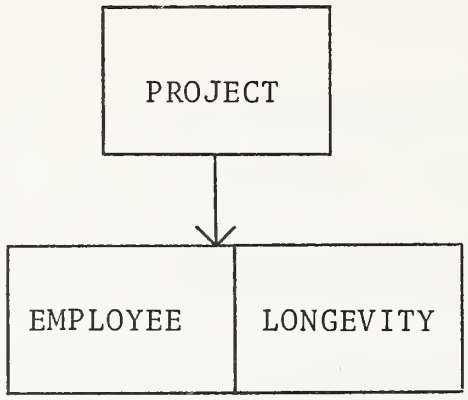
(2) EXPANDED.

(3) COMPRESSED.

FIGURE 4

OCCURRENCE DIAGRAMS TO ILLUSTRATE
IMPLEMENTING A ONE TO MANY RELATIONSHIP

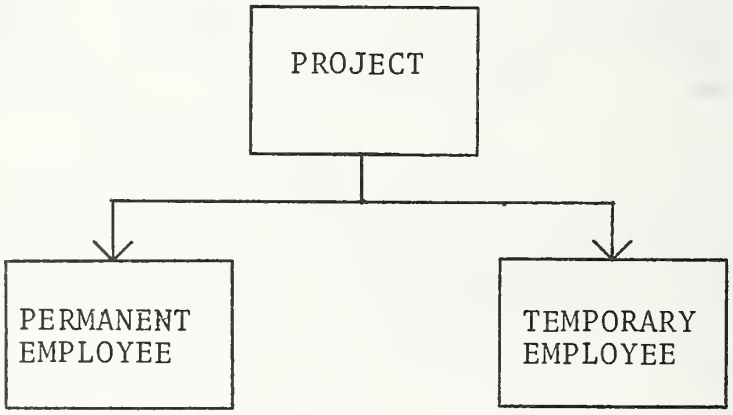
(1) MERGED.



PARENT

CHILD

(2)

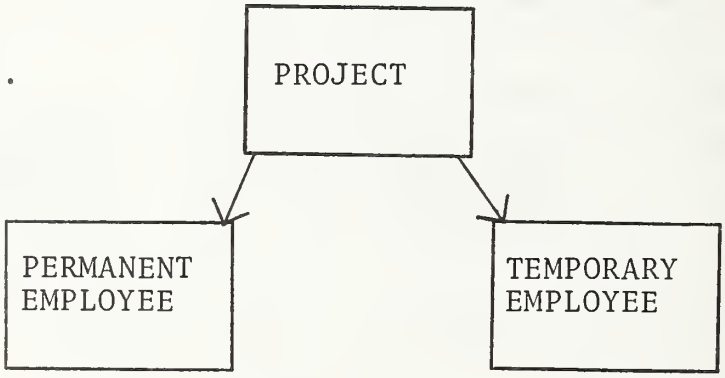


PARENT

CHILD

CHILD

(3) PARTITIONED.



PARENT

CHILD

CHILD

FIGURE 5

DATA STRUCTURE DIAGRAMS TO ILLUSTRATE
ISTINGUISHING TWO TYPES OF DESCENDANTS IN A ONE TO MANY RELATIONSHIP

merged. In the relational model, LONGEVITY would be an attribute in an employee relation.

- In the second diagram, two child record types (one for permanent employees and one for temporary employees) are members of the same set. In the relational model, similarly there would be two employee relations.

- In the third diagram, two child record types are members of different sets. The Data Translation Project calls this arrangement partitioned, while the entity-relationship model calls it split. In the relational model, again there would be two employee relations.

(3) Change string options associated with a one to many relationship. Included are:

- Changing between hierarchical pointers (which follow the depth-first hierarchical order of occurrences) and child/twin pointers (which indicate the first child and next twin) in IMS. These are shown in the occurrence diagrams in Figure 6, where we assume that PROJECT (e.g. tutorial writing) is not the root of a hierarchy.

- Adding or deleting back pointers. Examples include adding or deleting PRIOR pointers in CODASYL, as shown in the occurrence diagram in Figure 7, and changing between 2-way and 1-way pointers in IMS.

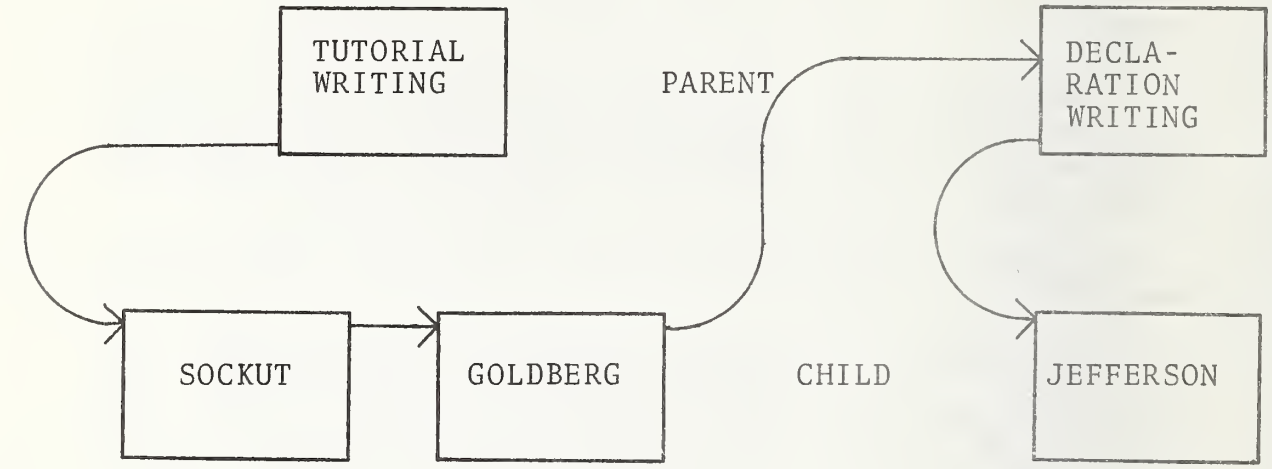
- Adding or deleting OWNER pointers in CODASYL, as shown in the occurrence diagram in Figure 7.

Since these options are implemented as pointer options in CODASYL and IMS, it may appear odd to classify them in the string level rather than in the encoding level. However, they change the direct access paths that can be followed in a relationship, and thus they belong at the string level.

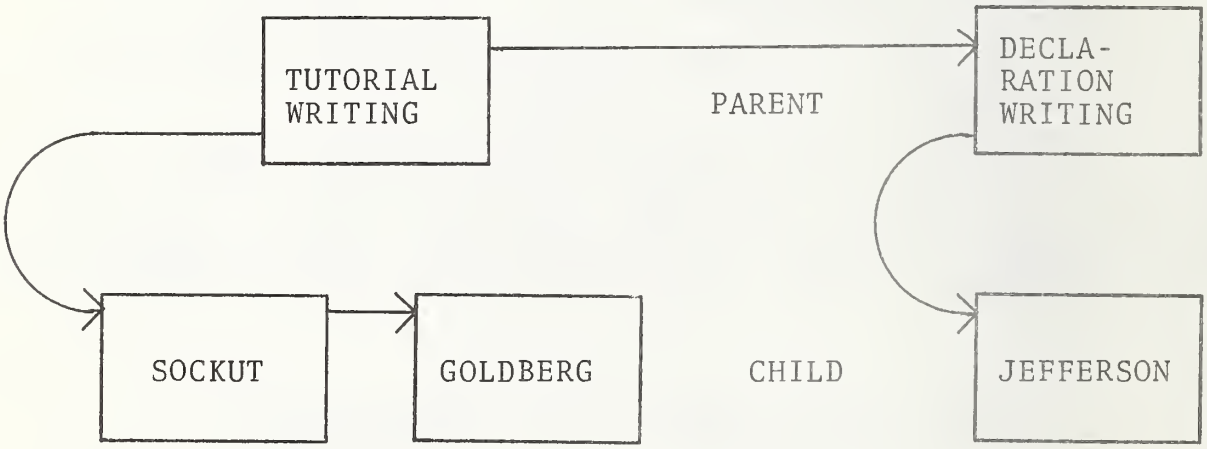
1.3.3 Other Changes at the String Level

A secondary index, hash scatter table, or CODASYL singular set (i.e. with OWNER = SYSTEM) can be created or destroyed. Such strings are used only as access paths, not as implementations of relationships.

The contents of strings can be changed. Examples are adding or removing a key in a densely indexed record type, physically duplicating or not duplicating a parent's field in its children (e.g. CODASYL ACTUAL SOURCE vs. VIRTUAL



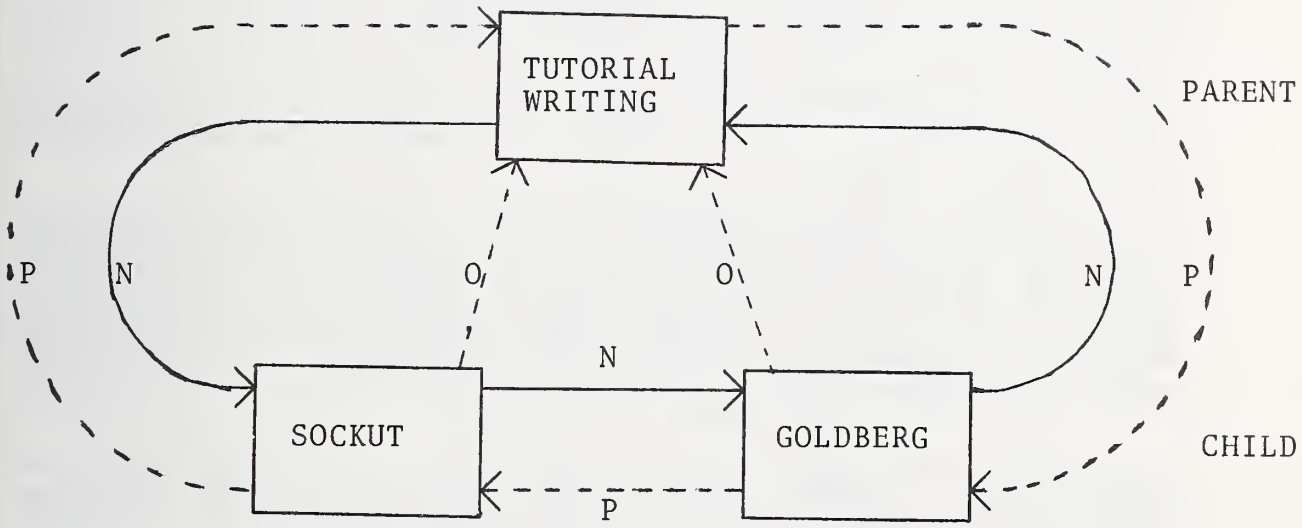
(1) HIERARCHICAL POINTERS.



(2) CHILD/TWIN POINTERS.

FIGURE 6

HIERARCHICAL VS. CHILD/TWIN POINTERS
IN AN IMS HIERARCHY OCCURRENCE



N ——→ NEXT (OR FIRST) POINTER
(REQUIRED, IF A POINTER CHAIN IS USED).

P - - - -→ PRIOR (OR LAST) POINTER (OPTIONAL).

O - - - -→ OWNER POINTER (OPTIONAL).

FIGURE 7

OPTIONAL POINTERS IN A CODASYL SET OCCURRENCE

SOURCE), and storing a field vs. calculating its value dynamically (e.g. CODASYL ACTUAL RESULT vs. VIRTUAL RESULT).

Ordering within strings can be changed. This includes reordering fields in a record type and changing the ordering strategy in a CODASYL set or IMS hierarchy. Examples of such changes in a set are changing the key(s) on which to order, changing between ascending and descending order, and changing among key ordered, order of creation, and unordered.

The changed security controls of a record type can be implemented (e.g. by a CODASYL PRIVACY LOCK).

1.4 Reorganization at the Encoding Level

In the examples below, string and attribute definitions are invariant. Only their physical representations (encodings) change.

Changes to relationship encoding definitions include changing between sequential (HSAM or HISAM) and direct (HDAM or HIDAM) organization in IMS and changing among embedded, array, and bit map pointers. These changes may also have an effect at the string level if they change the direct access paths that can be followed.

Below are examples of changing attribute encoding definitions. These changes are performed at the encoding level if they are to be invisible at the infological level. A DBMS may interpret encodings dynamically.

- Change basic representation (e.g. "APRIL" vs. "4").
- Change scale (e.g. inches vs. cm.; monthly vs. semi-monthly salary).
- Change character encoding (e.g. ASCII vs. EBCDIC).
- Change integer encoding (e.g. binary, packed decimal, decimal characters).
- Change field size (e.g. 16-bit integers).
- Change between fixed length and variable length.
- Change encryption.

- Change between data compression and non-compression [ALSB75]. Examples of data compression are eliminating leading zeros, eliminating trailing blanks, and encoding common data values. Data compression may require inclusion of a length indicator with the data.

1.5 Reorganization at the Physical Device Level

In this section's examples, representations of attributes and relationships are invariant. Only their physical placement changes.

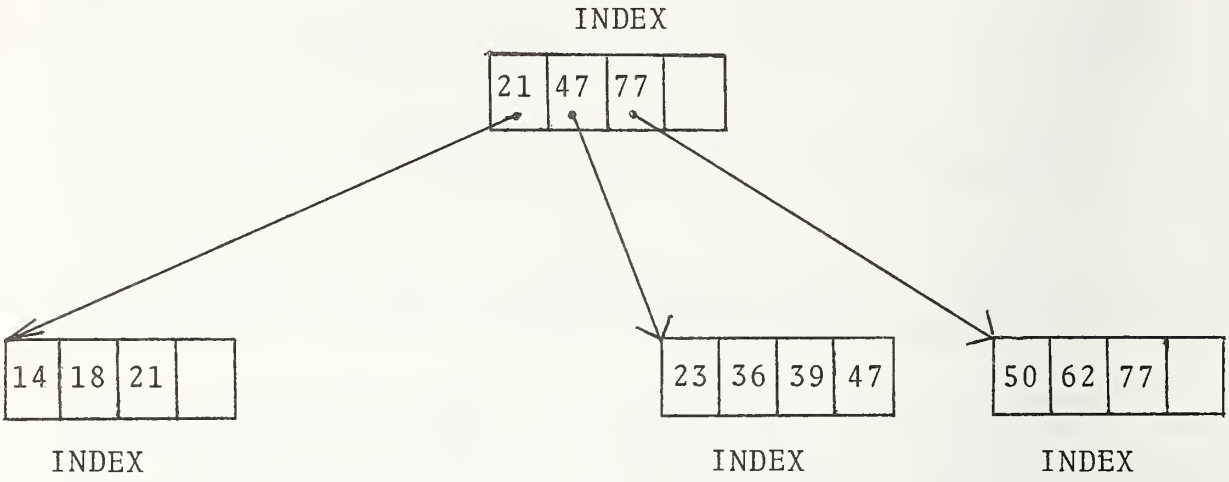
Changing frame definitions includes changing access keys and changing access methods. Examples of the latter include changing between CALC and VIA in CODASYL and changing between ISAM [IBM73] and VSAM [IBM76] (Virtual Storage Access Method), between HSAM and HISAM, or between HDAM and HIDAM in IMS.

Frame parameters can also be changed within an unchanged basic frame definition, e.g. to plan for growth or to improve performance. These parameters include the number of levels in an index hierarchy, the presence of an index table in CALC page headers, the distribution of free space for future insertions, the overflow / collision handling method, and hash parameters such as hashing function, page size, hash width (e.g. a modulus), and load factor.

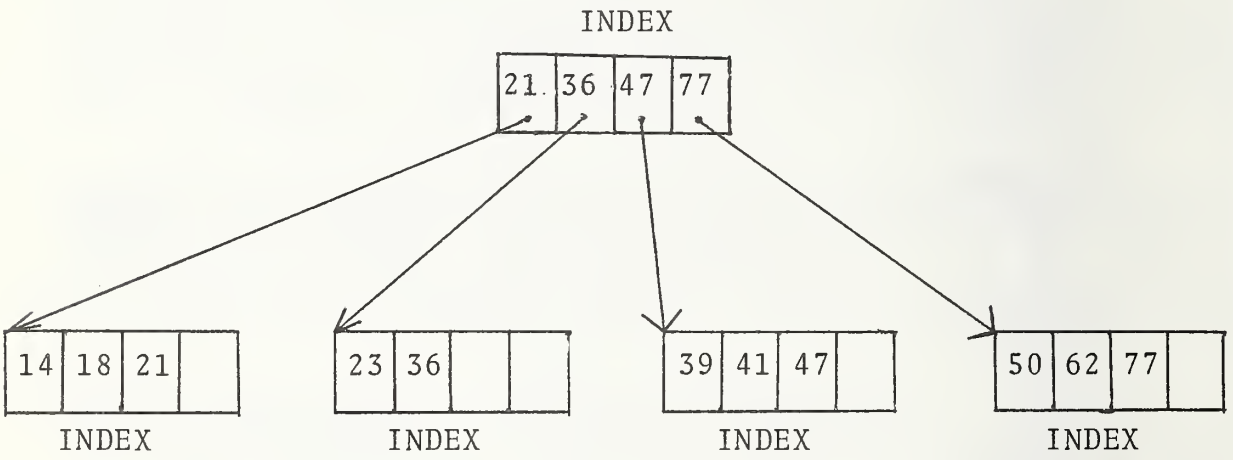
The mapping of the lowest level of frames to physical device subdivisions can be redefined. For example, CODASYL areas, VSAM Control Intervals, and VSAM Control Areas are mapped to tracks and cylinders. Also, portions of a data base can be moved to new storage devices permanently or temporarily.

Finally, maintenance operations (changes to frame occurrences) are performed repeatedly (periodically or upon demand) to improve access time and/or storage utilization. These operations are specific to the access method. In steady state (no growth), they may be performed only rarely (or never) for certain access methods. They are logically device-independent, but performance can be device-dependent, so an algorithm may be tailored to a specific device. Examples are listed below:

- Perform a split in VSAM, which is an access method that supports growth in an index hierarchy by splitting data areas (and, if necessary, indices) into two when required by insertion of data. An example is shown in Figure 8,



(1) PART OF INDEX HIERARCHY OCCURRENCE BEFORE INSERTION OF 41.



(2) PART OF INDEX HIERARCHY OCCURRENCE AFTER INSERTION OF 41.

FIGURE 8

A VSAM SPLIT

where the first diagram shows part of an index hierarchy occurrence in which we wish to insert an entry with key "41". There is no available space in the appropriate index (i.e. 23-47). Therefore VSAM splits this index into two, creates a new entry (36) in its parent index, and inserts the entry with key "41", as shown in the second diagram. If the parent index had had no available space for the new key and pointer, then it would have been split into two, and a new entry would have been inserted in its parent. Such propagation of splitting can continue to the level of the root index, which can be split if necessary (in which case the depth of the index hierarchy increases by 1).

- Eliminate or reduce overflow, e.g. remove ISAM overflow or move a hash synonym to its home slot if the record there is deleted. An example of the latter is shown in Figure 9. In the first diagram, record occurrence 1 is in its home hashing slot, while synonym records 2, 3, and 4 form a chain. After record 1 is deleted, one of the synonym records can be moved to the home slot, as shown for record 4 in the second diagram.

- In a linked list of unallocated areas, merge two contiguous areas into one larger area [ARME70].

- Balance an index hierarchy.

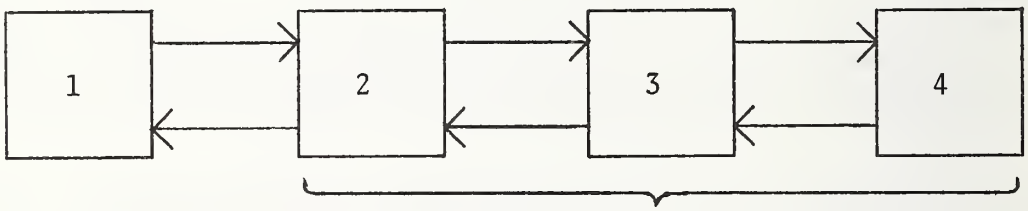
- Compact to make space occupied by deleted records contiguous.

- If physical order is logically unimportant, move frequently accessed objects to easily accessible positions.

- Make occurrences of physical proximity reflect occurrences of logical proximity. Examples are moving children closer to each other or to their parents and arranging a VSAM file so that the cylinder sequence reflects the key sequence (which may not be the case after a series of splits).

1.6 Other Terminology

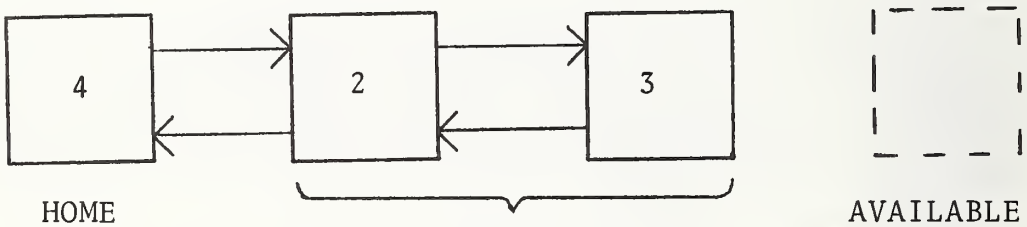
There is no universally used set of terms for types of reorganization. Several research groups interested in conversion and logical reorganization [SHU75; SHOS75; NAVA75] call logical reorganization restructuring and physical reorganization reformatting, and they use reorganization (as we do) as a generic term to cover both.



HOME SLOT

CHAIN OF HASH SYNONYMS

(1) BEFORE DELETION OF RECORD OCCURRENCE 1



HOME SLOT

CHAIN OF HASH SYNONYMS

AVAILABLE SPACE

(2) AFTER DELETION OF RECORD OCCURRENCE 1

FIGURE 9

OCCURRENCE DIAGRAM TO ILLUSTRATE
MOVING A HASH SYNONYM TO ITS HOME SLOT

Another dimension of classification is the distinction between one-shot and maintenance operations. One-shot operations (e.g. adding a secondary index) change definitions of constructs and generally are not planned to be performed repeatedly for a data base with stable characteristics. Maintenance operations (e.g. eliminating overflow in an indexed sequential access method) only change occurrences, not definitions, and they are performed repeatedly (either periodically or upon demand). Maintenance operations appear only at the lowest (physical device) level in the classification, while there are one-shot operations at all levels. Several researchers in maintenance [SHNE73, YA076, MARU76, TUEL78] use the term reorganization to denote what we call maintenance.

2. PRAGMATIC ISSUES

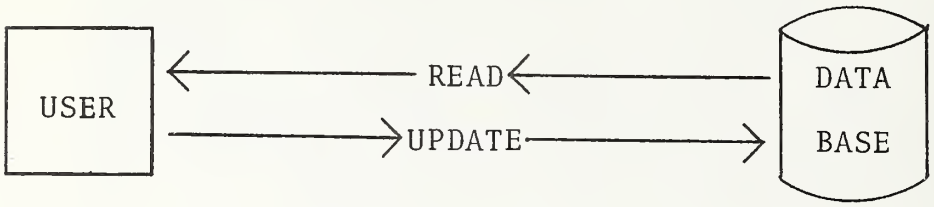
This section describes issues important to implementation and management of the types of reorganization described earlier. The issues include strategies for reorganization, commercial facilities, case studies, and data base administration considerations.

2.1 Strategies for Reorganization

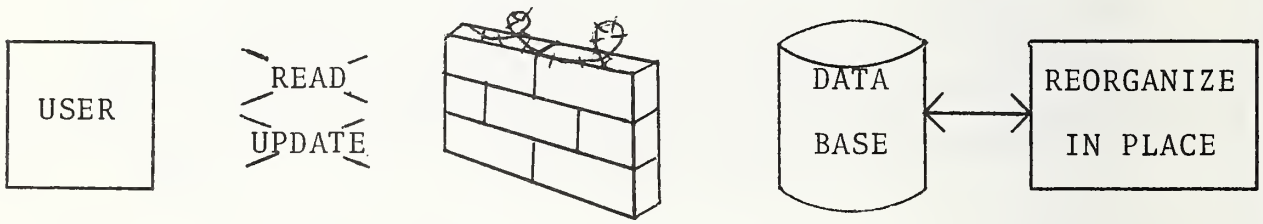
This section describes four strategies that can be used to reorganize a data base. The first three are commonly used in current DBMS, as shown in Section 2.2. The fourth appears to be used only for data base unloading in some DBMS. For the first two strategies, the data base, or at least the portion to be reorganized, is usually taken offline (i.e. is made unavailable for normal usage) overnight or over a weekend.

(1) In step 1 of reorganization in place (Figure 10), normal user access is allowed. In step 2, all user access is blocked while reorganization is performed in place. When reorganization is complete, normal access is allowed again, as in step 3. A variation on this strategy which can sometimes be used is merely to change the data base definition without performing physical reorganization.

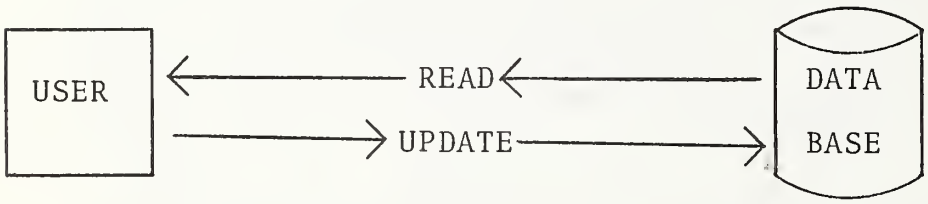
(2) In step 1 of reorganization by unloading and reloading (Figure 11), normal user access is allowed. All user access is blocked during step 2 (unloading onto an



STEP 1 NORMAL ACCESS



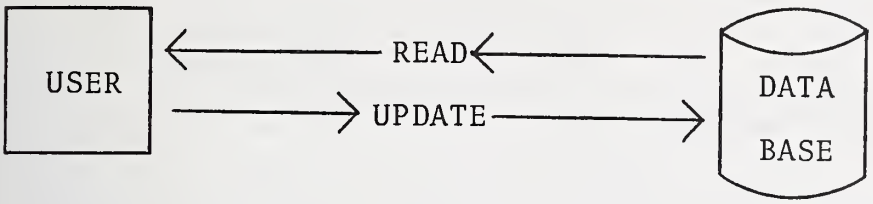
STEP 2 REORGANIZE IN PLACE



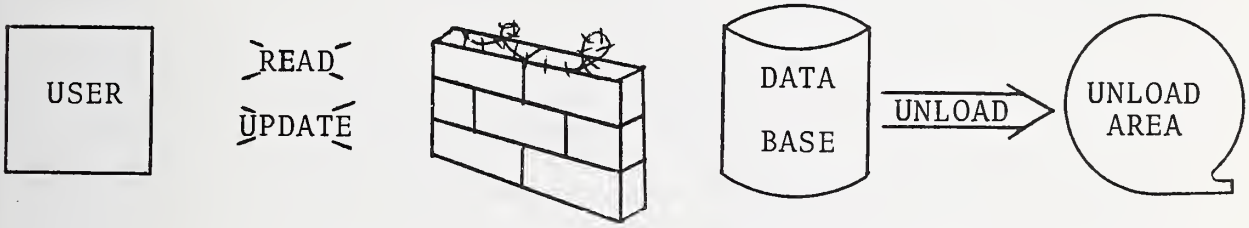
STEP 3 NORMAL ACCESS

FIGURE 10

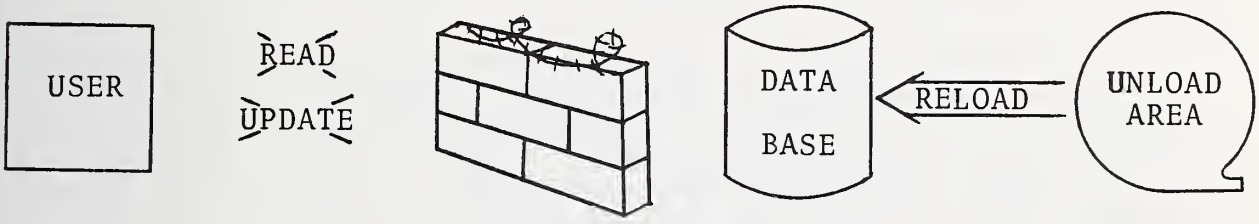
REORGANIZATION IN PLACE



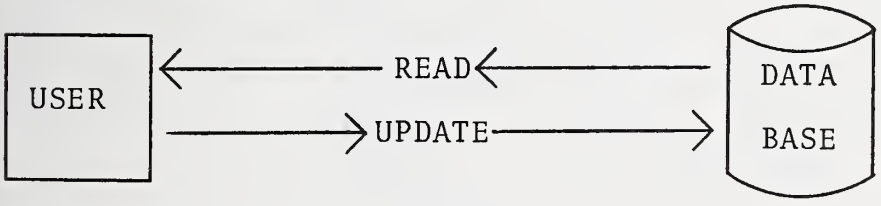
STEP 1 NORMAL ACCESS



STEP 2 UNLOAD



STEP 3 RELOAD AND REORGANIZE



STEP 4 NORMAL ACCESS

FIGURE 11

REORGANIZATION BY UNLOADING AND RELOADING

unload area) and step 3 (reloading in reorganized format). When reorganization is complete, normal access is allowed again, as in step 4. A variation on this strategy is to reorganize by copying from one area to another without using an intermediate unload area.

(3) A strategy that does not involve bringing the data base offline is incremental reorganization performed as objects are referenced. In this strategy, any needed reorganization occurs incrementally when a user references an object in the data base. An example is moving a hash synonym to its home slot when the record that was in the home slot is deleted.

(4) Another strategy that does not involve bringing the data base offline is reorganizing concurrently with usage of the data base. Under this strategy, users have access to the reorganized portion of the data base while one or more reorganization processes are reorganizing it (in place or by unloading and reloading), as pictured in Figure 12.

2.2 Commercial Facilities

Current commercial and special purpose DBMS usually provide facilities to perform some types of reorganization. The facilities employ the strategies described above. The number of applicable types of reorganization may vary from one system to another, but every system potentially requires some type. Below is a survey of functional capabilities of reorganization facilities provided with several well-known commercial DBMS: ADABAS, DMS 1100, IDMS, IMS, SYSTEM 2000, and TOTAL. For each system, some types of reorganization must be performed by customer-written programs that use the normal user interface. Most of the information was obtained from the vendors, not from first-hand experience. The survey does not compare the merits of the systems on the basis of their reorganization capabilities. The survey is intended to illustrate the current state of the art in types and methods of reorganization.

2.2.1 ADABAS

ADABAS [SOFT77] performs some types of maintenance incrementally as objects are referenced. When a record occurrence is deleted, that space within its physical block is recovered by compaction. As a file grows, filled blocks

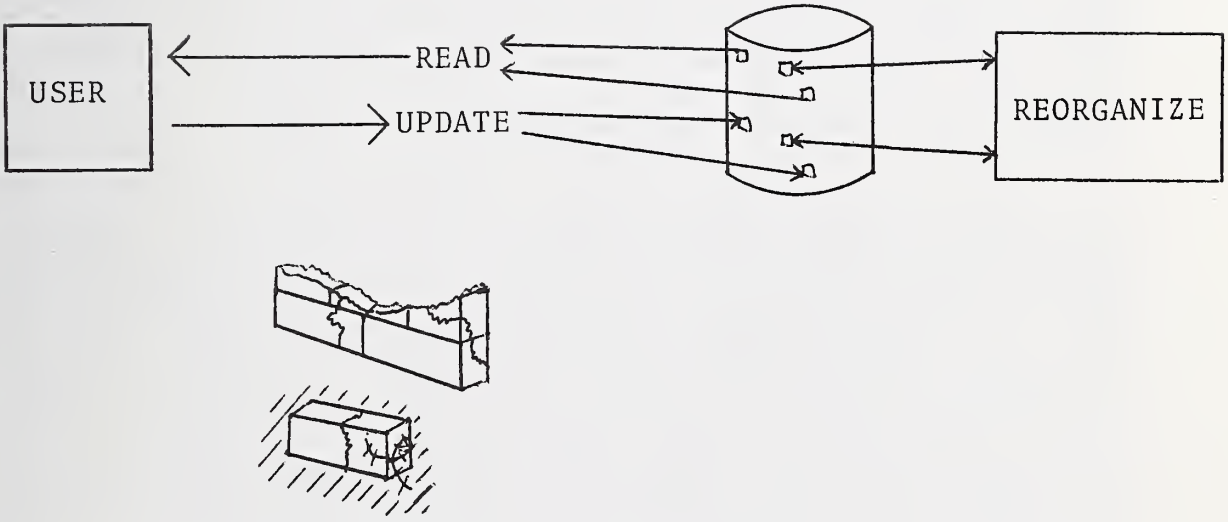


FIGURE 12

CONCURRENT REORGANIZATION AND USAGE

in its hierarchical index are split into two dynamically.

Security controls can be changed by changing the data base definition, without performing physical reorganization. Similarly, a field can be added (or deleted) at the end of a record type by just changing the record type's definition. When a record occurrence is then fetched, the field value is taken to be null until a non-null value is stored.

An unload and reload utility can reorder the data in a file or a portion of a file to make the physical order correspond to the logical order, to optimize performance of sequential processing. This utility can also be used to add or delete a field other than at the end of a record, to change between data compression and non-compression, and to change to or from hashed access. An unload and reload utility is also available to balance an index hierarchy without reorganizing the data.

Another utility can read a file and create a specified index, and a utility can read files and create a specified relationship by writing in a coupling structure. Most reorganization utilities lock at the file level, not at the whole data base level. Customer-written programs are needed to perform certain changes such as changing an attribute's representation or reordering fields, although such changes can be performed interpretively by a schema-subschema mapping.

2.2.2 DMS 1100

DMS 1100 is a CODASYL DBMS. A reorganization utility [SPER78] operates by unloading and reloading specified areas. Currently available functions include adding, deleting, or renaming areas, moving a record type from one area to another, rebuilding access linkages (hash chains, index sequential chains, and pointer arrays), moving set occurrence members closer to each other, moving displaced record occurrences back to their home pages, and changing parameters such as hashing functions, page load factors, page sizes, the number of pages, and overflow page distributions.

If a record type has been defined to include a large enough blank filler area at its end, then a new field can be added at the end by redefining the record type, without using a reorganization utility.

When a user deletes a record occurrence, the space is marked as deleted but is not immediately made available for reuse. If there is no room on a page for a new record occurrence, then the page is compacted in an attempt to avoid creating an overflow page. A utility is provided to operate in place to compact each page within a specified area.

Other types of reorganization must be performed by customer-written programs. Examples are changing an attribute's representation and changing from a one to one relationship to a one to many.

A reorganization utility [EDEL76] for DMS 1100 has been contributed to the user program library. Five steps are performed in using the utility:

(1) Unload selected record types.

(2) Delete them from the data base and compact their pages.

(3) Manipulate the unloaded records with a data editor (a processor for a programming language described below).

(4) Reload the records.

(5) Relink sets.

A user of the utility specifies old and new schemas and writes a data editor program that explicitly scans the old data and manipulates it from the old schema form into the new. The data editor's language includes variables, control, data manipulation facilities similar to CODASYL DML, and editing facilities similar to those of a general-purpose text editor. Some examples of types of reorganization that can be performed are changing a hashing function, changing a set's ordering criterion, changing a set between a chain and a pointer array, renaming or reordering fields in a record type, moving a record type from one area to another, changing the physical format of fields, changing between VIA and CALC, and adding or deleting record types, set types, fields, OWNER pointers, or PRIOR pointers. Some other types of reorganization, such as changing from a one to one relationship to a one to many or changing the representation of a field, cannot be performed easily, since it is difficult to manipulate individual fields. The data editor's language has approximately the same power as an ordinary application programming language, but it is oriented toward reorganization.

2.2.3 IDMS

IDMS is a CODASYL DBMS. A reorganization utility [CULL78] operates in place. Capabilities include adding, deleting, or reordering fields in a record type, changing all occurrences of a field to a constant value, changing the length of a field, adding or deleting a set, adding or deleting optional PRIOR or OWNER pointers in a set, changing a record type between fixed and variable length, and adding or deleting a database procedure for a record type (e.g. to change between data compression and non-compression). It is also possible to execute a data base procedure when the utility operates, e.g. to change an attribute's representation. New sets are initially empty. Record occurrences can be connected into them by executing an application program containing explicit connections (after reorganization has been completed).

An unload/reload utility permits changing an area's size and changing between hashing and indexing. Another utility can change page sizes by copying without using an intermediate unload file.

When a user deletes a record occurrence that is a member of a set without PRIOR pointers, the data is physically deleted, but the record's header is merely flagged as deleted. If that part of the set is later traversed in update mode, then the system unlinks the occurrence, and it is physically deleted at that time if it has been unlinked from all such sets. Utilities are also available to identify, unlink, and delete (physically) such flagged occurrences.

Some other types of maintenance are performed incrementally as objects are referenced: (1) when a record occurrence is physically deleted, that space within its page is recovered by compaction, and (2) if all or part of a record occurrence is moved off its home page when the record grows, then it can be moved back later when it is updated if space has become available.

If a record type has a large enough blank fill area at its end, then a new field can be added at the end by redefining the record type, without performing physical reorganization.

Some types of logical reorganization must be performed by customer-written programs. An example is changing from a one to one relationship to a one to many.

2.2.4 IMS

IMS provides utilities [IBM77b] that perform maintenance (recovering space and making the physical order reflect the hierarchical order) by unloading and reloading a file or a portion of a file. Some other types of reorganization can also be performed while using these utilities by specifying a new data base description for reloading. There can be no change in the hierarchical relationship of existing segment (record) types that remain in the reorganized data base. Changes that are allowed, subject to the above restriction, include deleting a segment type (if all occurrences have already been deleted), adding a segment type, changing pointer options, changing, adding, or deleting a non-key field in a segment type, changing a segment size, changing access methods, adding or deleting a secondary index, and changing a hash parameter. During unloading, users can read from the area being unloaded.

It is possible to add a new field at the end of a variable-length segment type by changing the segment type definition, without performing physical reorganization. When a segment occurrence is later read, its length field indicates whether a value has been stored in the new field in that segment occurrence.

Some forms of maintenance are performed incrementally as objects are referenced. Under HDAM and HIDAM (but not under HISAM), space occupied by a deleted segment occurrence is available for reuse. VSAM, which is often utilized with IMS, will support a growing hierarchical index by splitting data areas and indices into two dynamically, as shown in Figure 8 and described earlier.

Customer-written programs are needed for other types of reorganization, e.g. changing an attribute's representation, changing a relationship, and reordering fields.

2.2.5 SYSTEM 2000

SYSTEM 2000 [MRI78] can unload records and indices in logical order and then reload in physical order, which makes the logical and physical orders the same and makes available space contiguous. Noncontiguous available space may be created when record occurrences are deleted. The reload command can be directed to reload only those record occurrences that satisfy specified criteria. A record type or field is added or deleted by unloading and reloading with

a new data base definition if occurrences of the record type or a descendant record type exist. If no occurrences exist, then these operations are performed by just changing the data base definition, without performing physical reorganization.

A command is available to add or delete one or more indices, without reorganizing the actual data. An index can be rearranged to make its entries contiguous, without reorganizing the actual data. During these operations, users are blocked from only the affected portion of the data base.

Certain types of logical reorganization must be performed by customer-written programs. The user interface provides for performing an operation upon all occurrences of a record type, e.g. changing an attribute from monthly salary to semi-monthly salary.

2.2.6 TOTAL

TOTAL [CINC78] performs a type of maintenance incrementally when a record occurrence to be deleted is located in its home hashing slot: a hash synonym, if any (usually the physically most distant one), is moved to the home slot. When any record occurrence is deleted, the space that it (or a synonym) occupies is made available for reuse.

If a record type has been defined to include a large enough blank fill area at its end, e.g. for the purpose of adding new fields later, then a new field can be added at the end by redefining the record type, without performing physical reorganization. If there is no such fill area, or if a field is to be added other than at the end, then unload and reload utilities are used. Unload and reload utilities are also used to move record occurrences closer together, to move a field from one record type to another, to change relationships, and to change a file's size (at which time the hash width may be changed). During unloading, users can read from the area being unloaded. A new record type and its relationships can be added without unloading and reloading existing record types if the old record types have been defined to include enough blank space for new pointers.

An in-place reorganization utility is available to change all occurrences of a field from one set of specific values to another, e.g. from "APRIL" to "4", but customer-written programs are needed to perform more general field changes, e.g. to halve all occurrences to represent

changing from monthly to semi-monthly salaries.

2.3 Case Studies

Statistics were gathered for several installations regarding the types of operations performed, the amount of time and effort required, and the strategy and facilities used to reorganize. One very large data base [DARD77] occupies 22 disk packs, each having a capacity of 200 megabytes. At this installation, reorganization is performed by unloading and reloading the entire data base. Users can read (but not update) during unloading, and no access is allowed during reloading. The elapsed time required to reload is approximately 40 hours. At this installation, reloading cannot be continued if an error occurs. On several reloads there have been failures, e.g. due to tape I/O errors. Those reloads had to be restarted from the beginning.

Another case study involved the FBI's National Crime Information Center [BUEL77, WEIS78], which is to be available 24 hours per day if possible. It maintains up-to-date information on various crimes, such as car theft. It does not use a commercial DBMS, and there are no relationships among files, but there are similarities to DBMS. Most files use ISAM [IBM73]. Every two weeks, maintenance is performed in 2 steps:

(1) Copy (and reorganize) from disk to disk. This requires at least 6 hours. Users can read or delete records (from the old files) but cannot perform any other types of updates. The system keeps track of deletions.

(2) Replace the old files by the reorganized files. This requires approximately 45 minutes. All user access is blocked. Deletions that were performed in the old files during step 1 are now performed in the reorganized files. It was necessary to allow such deletions in order to prevent situations in which, for example, a stolen car is recovered during a reorganization period, the owner then drives it from the police station, and the owner is mistakenly arrested shortly thereafter because the stolen car record has not yet been deleted.

Approximately 4 times per year, other types of reorganization are performed via the same copying strategy. These have included adding new fields, changing the locations of fields, adding and deleting secondary indices, changing the access method used for secondary indices,

changing the number of levels in index hierarchies, and changing the locations of master indices.

The final case study involved an airline reservation system [SIWI77], which is to be available 24 hours per day if possible. It is not a general-purpose DBMS. An installation's initial logical structure is designed carefully, and logical changes are rarely made later. At least 3 types of reorganization have been performed:

- Occasionally (approximately every 2 years), a record type is added or a file's size is expanded. Reorganization is performed by unloading all the disks and then reloading them. During unloading, users can read from the data base, and updates are saved in a special area. The updates are later written into the data base. No user access is allowed during reloading. Unloading and reloading each require 5-10 minutes per disk pack. There are typically 2 to 200 disk packs per installation.

- At one installation, the data base was transferred from one set of disks to a faster set of disks, and it was moved geographically at the same time. This required approximately 24 hours, during which users were allowed to read from the old area, while updates were saved in a special area. The updates were later written into the new area. The careful planning and the writing of special programs that were used required several months.

- Maintenance is usually performed every night. This involves purging records for flights that have already flown. It also includes construction of an index, as follows: at any time, flights for the next n days are stored with a fine index for quick access, while flights for later days are stored with a coarse index to save space. During maintenance, fine indices are constructed for the flights for the nth day. The granularity of locking during maintenance is fine enough so that a user transaction is blocked for only a few seconds.

2.4 Data Base Administration Considerations

Reorganization of a data base such as those described above is usually managed by the data base administrator (DBA) [MELT75, LYON76, LEON78, FRY78]. The DBA is the individual (or group of individuals) responsible for logical and physical definition of the data base, setting security and integrity policies, monitoring performance, and, in general, supervising use of the data base. The DBA

determines that reorganization should be performed. The DBA must consider the following issues in connection with reorganization:

- Recognize the need to reorganize. Examples of reasons for reorganization were described earlier.

- Decide what new structures are to be the final result of reorganization. Some examples are a new hash placement, balanced index hierarchies, or a new set.

- Decide when to perform reorganization. For maintenance, there may be an optimal period between reorganizations. For any reorganization, it may be necessary to reorganize overnight or over a weekend.

- Know how to execute the reorganization.

  -- Strategies may involve unloading and reloading, reorganizing in place but offline, reorganizing incrementally as objects are referenced, or reorganizing concurrently with usage.

  -- The DBA must select the appropriate reorganization facilities.

  -- Journalling facilities are useful to recover from errors, if re-execution of reorganization would take longer than recovery would.

- Determine how much the reorganization will benefit the enterprise. This may include improved performance, increased functional capabilities, or better storage utilization. In the case of reorganization to improve performance, performance prediction tools [e.g. BUZE78, DEUT78] are useful.

- Assess how much reorganization will cost. This includes:

  -- Human and computational resources consumed during planning, actual reorganization, software changes (if any), and personnel retraining (if any).

  -- Either the denial of resources to users during offline reorganization or degraded user performance during concurrent reorganization.

- Be aware of who and what will be affected by reorganization. For example, the first phase of at least one reorganization utility [SPER78] analyzes a proposed reorganization and lists the affected portions of the data base. Another tool which often can determine

effects of reorganization is a data dictionary / directory [LEON77], which maintains information such as what structures are in the data base and which applications use them. Some applications may benefit from the reorganization, while others may suffer if the data base is no longer optimized toward them. The DBA must act as arbitrator. The DBA must also see that affected software is revised and that personnel retraining is provided for any affected users.

- Document any changes that result from reorganization. Some of this documentation may be provided by the data dictionary / directory.

- Certify that reorganization yielded the desired result. For example, this can involve checking that new pointers correctly implement a new relationship.

3. RESEARCH EFFORTS

We have seen that many situations require reorganization, and that reorganization requires time and effort. In this section we survey several research efforts in three aspects of reorganization: (1) conversion, (2) maintenance, and (3) concurrent reorganization and usage. Concepts that were studied may become future DBMS trends.

3.1 Conversion

Several research groups, e.g. those at the University of Pennsylvania [SMIT71, RAMI74], the University of Michigan [SIBL73, FRY74, LEWI75, NAVA75, SWAR77], the University of Florida [SU74, NATI78], the IBM San Jose research laboratory [SHU75, SHU77], and System Development Corp. [SHOS75], have been developing semantics and languages for specification of logical reorganization. These languages are to be used during data base conversion, e.g. from one DBMS's definition (a source) to another's (a target), using a procedure such as that shown in Figure 13. The DBA defines old and new structures in a non-procedural data description language and defines their correspondence in a non-procedural data translation (logical reorganization) language. Data translation language statements indicate how source structures are mapped into target structures. A reader, which is driven by the data description language statements

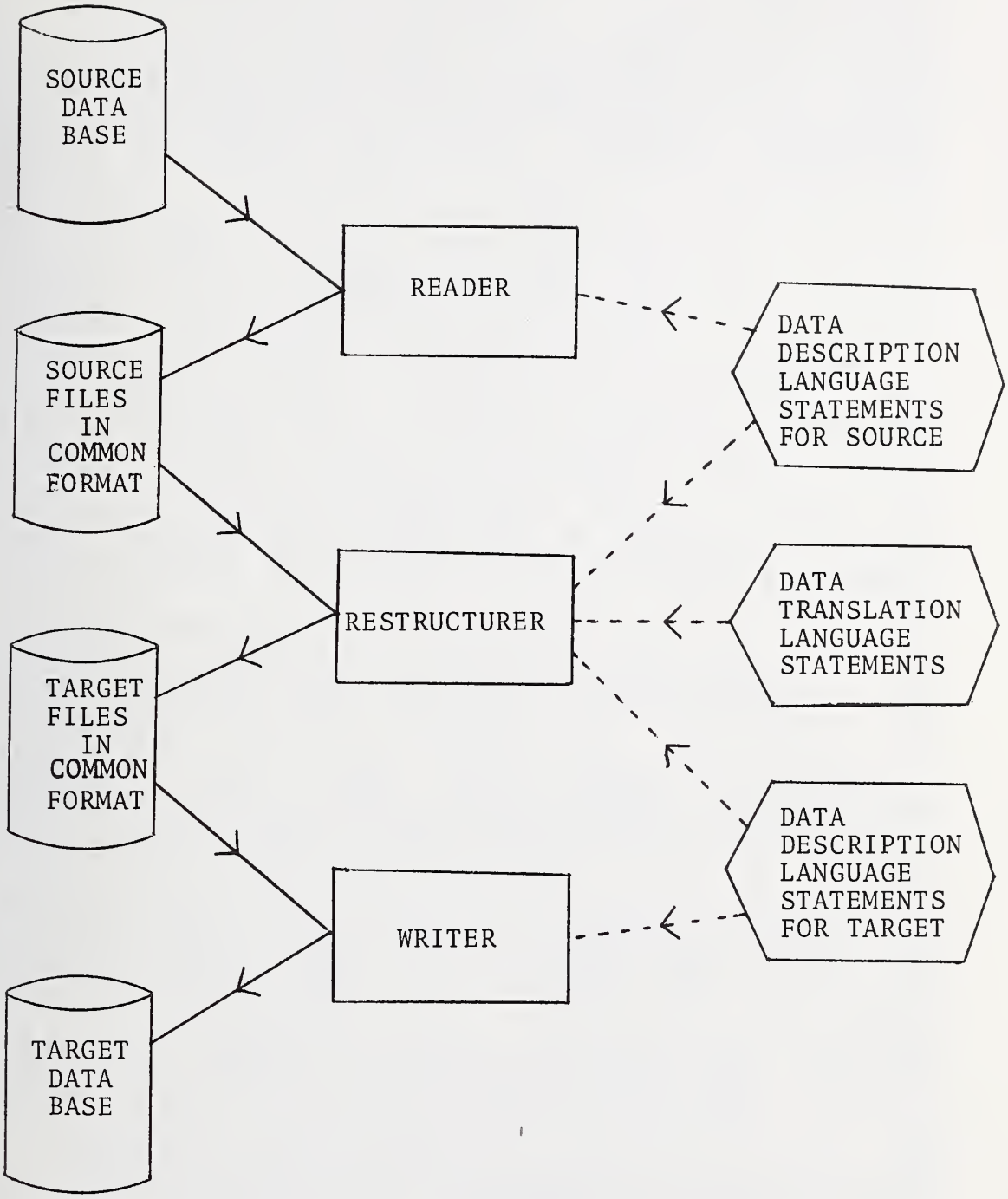


FIGURE 13

A DATA TRANSLATION PROCEDURE

for the source, converts the source into a common format for reorganization. A restructurer, which is driven by the data translation language statements and by both sets of data description language statements, produces a translated target file in common format. Finally, a writer, which is driven by the data description language statements for the target, converts the target file into its final format. The translation procedure might be compiled instead of interpreted.

3.2 Maintenance

Several authors [e.g. SHNE73, YAO76, MARU76, TUEL78] have modeled data base performance deterioration, improvement through maintenance, and maintenance costs. Factors in the analyses include rates of growth and updates, access times for various structures as a function of growth and updates, access times after maintenance is performed, and the time required to perform maintenance. Results include strategies for determining when to perform maintenance.

3.3 Concurrent Reorganization and Usage

Another research area deals with performing reorganization concurrently with data base usage. Offline reorganization is unsatisfactory for two classes of data bases: (1) If an essential computer utility, such as a military, hospital, police, or reservation data base, is to be available 24 hours per day, then it cannot be brought offline for significant periods of time. (2) For a very large data base, such as census data, reorganization might require much longer than a holiday weekend. "A very large database is a database whose reorganization by reloading takes a longer time than the users can afford to have the database unavailable" [WIED77, p. 449]. Also, many DBAs prefer 24-hour availability, even if it is not essential. Thus it is appropriate, and increasingly necessary, to use techniques such as reorganizing concurrently with full usage of the reorganized portion.

Several research studies have dealt with this area. One study [SOCK77] produced requirements for concurrent reorganization [SOCK76] (e.g. locking), the classification of types of reorganization described earlier, and a performance model [SOCK78]. The model predicts degradation

of user response time and reorganization time under concurrent reorganization and usage. Parameters include user access characteristics and reorganization characteristics. Another study [WILS79] produced a set of operational rules to ensure correctness of concurrent reorganization and usage of CODASYL data bases. A prototype implementation can add, delete, and reorder fields within a record type. An alternative to in-place concurrent reorganization is recording attempted user updates during unloading and reloading and later executing those updates. SEVE76 describes a technique for managing this strategy and determining which objects have been updated. Another alternative is reorganizing incrementally as objects are referenced. GERR76 describes such a strategy for managing co-existing "generations" of schemas.

CONCLUSIONS

The following conclusions can be drawn from our study:

- Data base reorganization must be considered a necessary function. Failure to reorganize can result in high expense (as extra space and time are consumed), user dissatisfaction (as response times increase), and limitations on functional capabilities (as desired new information cannot be represented). All data base systems potentially require some type of reorganization, and not all future instances of reorganization can be predicted when a data base is created.

- Many varieties of reorganization exist, e.g. overflow removal or addition of a relationship. Current DBMS generally provide facilities to perform many but not all types of reorganization.

- Performing reorganization can be time-consuming and hence expensive, especially in very large data bases. This can be intolerable in essential 24-hour utilities.

- An installation must have a reorganization policy. The data base administrator must determine this policy, since reorganization can affect all users.

- Several research efforts in reorganization have been undertaken. Results of this research will increase in importance as very large data bases become more common.

ACKNOWLEDGEMENTS

The authors would like to thank Deborah A. Sheetz, David K. Hsiao, Stuart E. Madnick, Donald R. Deutsch, John L. Berg, Belkis W. Leong-Hong, and Peter Mager for reviewing an earlier draft of this paper. Peter P.-S. Chen, Ugo O. Gagliardi, Michael E. Senko, and Herbert S. Meltzer reviewed related earlier work. Several vendor representatives clarified our understanding of their systems.

REFERENCES

[ALSB75] Alsberg, P. A. "Space and time savings through large data base compression and dynamic restructuring", Proc. IEEE 63, 8 (Aug. 1975), 1114-1122.

[ALTM72] Altman, E. B.; Astrahan, M. M.; Fehder, P. L.; and Senko, M. E. "Specifications in a data independent accessing model", Proc. ACM SIGFIDET Workshop on Data Description, Access, and Control, Nov. 1972, pp. 363-382.

[ARME70] Armenti, A. W.; Galley, S. W.; Goldberg, R. P.; Nolan, J. F.; and Sholl, A. "LISTAR - Lincoln information storage and associative retrieval system", Proc. Spr. Jt. Computer. Conf., vol. 36, AFIPS, May 1970, pp. 313-322.

[ASTR72] Astrahan, M. M.; Altman, E. B.; Fehder, P. L.; and Senko, M. E. "Concepts of a data independent accessing model", Proc. ACM SIGFIDET Workshop on Data Description, Access, and Control, Nov. 1972, pp. 349-362.

[BACH69] Bachman, C. W. "Data structure diagrams", Data Base (ACM SIGBDP) 1, 2 (Summer 1969), 4-10.

[BUEL77] Buell, F. B. private commun., Mar. 1977.

[BUZE78] Buzen, J. P.; Goldberg, R. P.; Langer, A. M.; Lentz, E.; Schwenk, H. S., Jr.; Sheetz, D. A.; and Shum, A. W.-C. "BEST/1 - Design of a tool for computer system capacity planning", Proc. Natl. Computer Conf., vol. 47, AFIPS, June 1978, pp. 447-455.

[CHEN78] Chen, P. P.-S. "The Entity-relationship approach to logical data base design", Data Base Monograph #6, Q.E.D. Information Sciences, Inc., Wellesley, MA, 1978.

[CHEN77] Chen, P. P.-S. "The Entity-relationship model - a basis for the enterprise view of data", Proc. Natl. Computer Conf., vol. 46, AFIPS, June 1977, pp. 77-84.

[CHEN75] Chen, P. P.-S. "The Entity-relationship model -- toward a unified view of data", Proc. [First] Int. Conf. on Very Large Data Bases (avail. from ACM), Sept. 1975, p. 173 (abstract); ACM Trans. Database Syst. 1, 1 (Mar. 1976), 9-36.

[CINC78] Cincom Systems, Inc., "TOTAL/8 data base administration reference manual", Apr. 1978.

[CODA78] CODASYL Data Description Language Committee, "Journal of development" (avail. from Materiel Data Management Branch, Dept. of Supply and Services, Canadian govt., Hull, Que., Canada), Jan. 1978.

[CODA71] CODASYL Programming Language Committee, "Data base task group report" (avail. from ACM), Apr. 1971.

[CODA77] CODASYL Systems Committee, Stored-Data Definition and Translation Task Group, "Stored-data description and data translation: a model and language", Information Syst. 2, 3 (1977), 95-148.

[CODD70] Codd, E. F. "A Relational model of data for large shared data banks", Commun. ACM 13, 6 (June 1970), 377-387.

[CULL78] Cullinane Corp., "IDMS utilities", release 5.0, Sept. 1978.

[DARD77] Dardwin, D. G. "Reloading tough work with huge data base", Computerworld 11, 45 (Nov. 7, 1977), 38.

[DATE77] Date, C. J. An Introduction to database systems, second ed., Addison-Wesley, Reading, MA, 1977.

[DEUT78] Deutsch, D. R. "Modeling and measurement techniques for the evaluation of design alternatives in the implementation of database management software", D.B.A. diss., Coll. of Bus. and Management, Univ. of Maryland, College Park, MD, Dec. 1978; Nat. Bur. of Standards special pub., 1979.

[EDEL76] Edelman, J. A.; Liaw, Y. S.; Nazif, Z. A.; and Scheidt, D. L. "Reorganization system user's reference manual", USE Program Library Interchange, Sperry Univac, Oct. 1976.

[FRY74] Fry, J. P.; and Jeris, D. W. "Towards a formulation and definition of data reorganization", Proc. ACM SIGMOD Workshop on Data Description, Access and Control, May 1974, pp. 83-100.

[FRY78] Fry, J. P.; Teorey, T. J.; DeSmith, D. A.; and Oberlander, L. B. "Survey of state-of-the-art database administration tools: survey results and evaluation", Database Syst. Research Group Tech. Rep. DSRG 78 DE 14.2, Grad. School of Bus. Admin., Univ. of Michigan, Ann Arbor, MI, Aug. 1978.

[GERR76] Gerritsen, R.; and Morgan, H. L. "Dynamic restructuring of databases with generation data structures", Proc. ACM Annual Conf., Oct. 1976, pp. 281-286.

[HOUS77] Housel, B. C. "A Unified approach to program and data conversion", Proc. Third Int. Conf. on Very Large Data Bases (avail. from ACM), Oct. 1977, pp. 327-335.

[IBM77a] IBM Corp., "Information Management System / Virtual Storage (IMS/VS) general information manual", form GH20-1260-6, July 1977.

[IBM77b] IBM Corp., "Information Management System / Virtual Storage (IMS/VS) utilities reference manual", form SH20-9029-4, July 1977.

[IBM76] IBM Corp., "OS/VS Virtual Storage Access Method (VSAM) programmer's guide", form GC20-3838-2, Apr. 1976.

[IBM73] IBM Corp., "OS/360 data management services guide", form GC26-3746-2, July 1973.

[KERS76] Kerschberg, L.; Klug, A. C.; and Tsichritzis, D. C. "A Taxonomy of data models", in Lockemann, P. C.; and Neuhold, E. J. (Eds.). Syst. for Large Data Bases (Proc. Second Int. Conf. on Very Large Data Bases, Sept. 1976), North-Holland, Amsterdam, Neth., 1977, pp. 43-64.

[LEON78] Leong-Hong, B. W.; and Marron, B. "Database administration: concepts, tools, experiences, and problems", Nat. Bur. of Standards special pub. 500-28, Mar. 1978.

[LEON77] Leong-Hong, B. W.; and Marron, B. "Technical profile of seven data element dictionary / directory systems", Nat. Bur. of Standards special pub. 500-3, Feb. 1977.

[LEWI75] Lewis, K.; Driver, B.; and Deppe, M. E. "A Translation definition language for the version II translator", Data Translation Proj. working paper 809, Grad. School of Bus. Admin., Univ. of Michigan, Ann Arbor, MI, Apr. 1975.

[LYON76] Lyon, J. K. The Database administrator, John Wiley & Sons, New York, NY, 1976.

[MART77] Martin, J. Computer data-base organization, second ed., Prentice-Hall, Englewood Cliffs, NJ, 1977.

[MARU76] Maruyama, K.; and Smith, S. E. "Optimal reorganization of distributed space disk files", Commun. ACM 19, 11 (Nov. 1976), 634-642.

[MELT75] Meltzer, H. S. "An Overview of the administration of data bases", Proc. Second USA - Japan Computer Conf., AFIPS, Aug. 1975, pp. 365-370.

[MRI78] MRI Systems Corp., "System 2000 general information manual", 1978.

[NATI78] Nations, J.; and Su, S. Y. W. "Some DML instruction sequences for application program analysis and conversion", Proc. ACM SIGMOD Int. Conf. on Management of Data, May 1978, pp. 120-131.

[NAVA75] Navathe, S. B.; and Fry, J. P. "Restructuring for large data bases: Three levels of abstraction", Proc. [First] Int. Conf. on Very Large Data Bases (avail. from ACM), Sept. 1975, p. 174 (abstract); ACM Trans. Database Syst. 1, 2 (June 1976), 138-158.

[RAMI74] Ramirez, J. A.; Rin, N. A.; and Prywes, N. S. "Automatic generation of data conversion programs using a data description language", Proc. ACM SIGMOD Workshop on Data Description, Access, and Control, May 1974, pp. 207-225.

[SCHN76] Schneider, L. S. "A Relational view of the data independent accessing model", Proc. ACM SIGMOD Int. Conf. on Management of Data, June 1976, pp. 75-90.

[SENK75] Senko, M. E. "Specification of stored data structures and desired output results in DIAM II with FORAL", Proc. [First] Int. Conf. on Very Large Data Bases (avail. from ACM), Sept. 1975, pp. 557-571.

[SENK76] Senko, M. E.; and Altman, E. B. "DIAM II and levels of abstraction. The Physical device level: A General model for access methods", in Lockemann, P. C.; and Neuhold, E. J. (Eds.). Syst. for Large Data Bases (Proc. Second Int. Conf. on Very Large Data Bases, Sept. 1976), North-Holland, Amsterdam, Neth., 1977, pp. 79-94.

[SEVE76] Severance, D. G.; and Lohman, G. M. "Differential files: their application to the maintenance of large databases", Proc. ACM SIGMOD Int. Conf. on Management of Data, June 1976, p. 43 (abstract); ACM Trans. Database Syst. 1, 3 (Sept. 1976), 256-267.

[SHNE73] Shneiderman, B. "Optimum data base reorganization points", Commun. ACM 16, 6 (June 1973), 362-365.

[SHOS75] Shoshani, A. "A Logical-level approach to data base conversion", Proc. ACM SIGMOD Int. Conf. on Management of Data, May 1975, pp. 112-122.

[SHU75] Shu, N. C.; Housel, B. C.; and Lum, V. Y. "CONVERT: A High level translation definition language for data conversion", Proc. ACM SIGMOD Int. Conf. on Management of Data, May 1975, p. 111 (abstract); Commun. ACM 18, 10 (Oct. 1975), 557-567.

[SHU77] Shu, N. C.; Housel, B. C.; Taylor, R. W.; Ghosh, S. P.; and Lum, V. Y. "EXPRESS: A Data EXtraction, Processing, and REStructuring System", ACM Trans. Database Syst. 2, 2 (June 1977), 134-174.

[SIBL76] Sibley, E. H. (Ed.). Comput. Surv. (special issue on data base management syst.) 8, 1 (Mar. 1976).

[SIBL73] Sibley, E. H.; and Taylor, R. W. "A Data definition and mapping language", Commun. ACM 16, 12 (Dec. 1973), 750-759.

[SIWI77] Siwiec, J. E. "A High-performance DB/DC system", IBM Syst. J. 16, 2 (1977), 169-195; and private commun., June 1977.

[SMIT71] Smith, D. C. P. "An Approach to data description and conversion", Ph.D. diss., Moore School of Elec. Engin., Univ. of Pennsylvania, Philadelphia, PA, 1971.

[SOCK78] Sockut, G. H. "A Performance model for computer data-base reorganization performed concurrently with usage", Operations Res. 26, 5 (Sept.-Oct. 1978), 789-804.

[SOCK77] Sockut, G. H. "Data base performance under concurrent reorganization and usage", Ph.D. thesis, Div. of Applied Sciences, Harvard Univ., Cambridge, MA, Nov. 1977; Center for Research in Computing Technology Tech. Rep. 12-77, Harvard Univ., July 1977.

[SOCK76] Sockut, G. H.; and Goldberg, R. P. "Motivation for data base reorganization performed concurrently with usage", Center for Research in Computing Technology Tech. Rep. 16-76, Harvard Univ., Cambridge, MA, Sept. 1976; preprints, ACM Computer Science Conf., Jan. 1977, p. 26 (abstract); Data Base Engin. (IEEE-CS Tech. Comm. on Data Base Engin.) (preprints, IEEE-CS Workshop on Operating and Data Base Management Syst., Mar. 1977) 1, 1 (Mar. 1977), 18-19 (abstract).

[SOFT77] Software AG of North America, Inc., "ADABAS introduction", 1977.

[SPER78] Sperry Univac, "Data Management System (DMS 1100) level 8R1 system support functions data administrator reference", Univac pub. UP 7909.1, 1978.

[SU74] Su, S. Y. W.; and Lam, H. "A Semi-automatic data base translation system for achieving data sharing in a network environment", Proc. ACM SIGMOD Workshop on Data Description, Access, and Control, May 1974, pp. 227-247.

[SWAR77] Swartwout, D. E.; Deppe, M. E.; and Fry, J. P. "Operational software for restructuring network databases", Proc. Natl. Computer Conf., vol. 46, AFIPS, June 1977, pp. 499-508.

[TSIC77] Tsichritzis, D. C.; and Klug, A. C. (Eds.). "ANSI/X3/SPARC DBMS framework report of the study group on data base management systems", AFIPS, Nov. 1977.

[TUEL78] Tuel, W. G., Jr. "Optimum reorganization points for linearly growing files", ACM Trans. Database Syst. 3, 1 (Mar. 1978), 32-40.

[WEIS78] Weise, R. private commun., Aug. 1978.

[WIED77] Wiederhold, G. Database design, McGraw-Hill, New York, NY, 1977.

[WILS79] Wilson, T. B. "A General model for dynamic data base restructuring" (avail. from author at Sperry Univac, Roseville, MN), 1979.

[YAO76] Yao, S. B.; Das, K. S.; and Teorey, T. J. "A Dynamic database reorganization algorithm", ACM Trans. Database Syst. 1, 2 (June 1976), 159-174.

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. NBS SP 500-47	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE

Data Base Reorganization -- Principles and Practice

5. Publication Date

April 1979

6. Performing Organization Code

7. AUTHOR(S)

Gary H. Sockut and Robert P. Goldberg

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)*

same as item 9

13. Type of Report & Period Covered

Final

14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 79-600055

☐ 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.)*

Data base reorganization can be defined as changing some aspect of the way in which a data base is arranged logically and/or physically. This paper contains tutorials and surveys. It introduces the basic concepts of reorganization, including why it is performed. Many examples of types of reorganization are described and are classified into logical / physical levels. The paper then covers pragmatic issues such as reorganization strategies, a survey of several commercial reorganization facilities, case studies, and data base administration considerations. Finally, several research efforts are surveyed.

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)*

Data base; data base management;
file maintenance; reformatting; reorganization; restructuring.

18. AVAILABILITY ☒ Unlimited

☐ For Official Distribution. Do Not Release to NTIS

☐ Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402, SD Stock No. SN003-003-02055-9

☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161

19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PAGES
UNCLASSIFIED	51
20. SECURITY CLASS (THIS PAGE)	**22. Price**
UNCLASSIFIED	$2.30

USCOMM-DC

☆U.S. GOVERNMENT PRINTING OFFICE: 1979-281-067/100

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

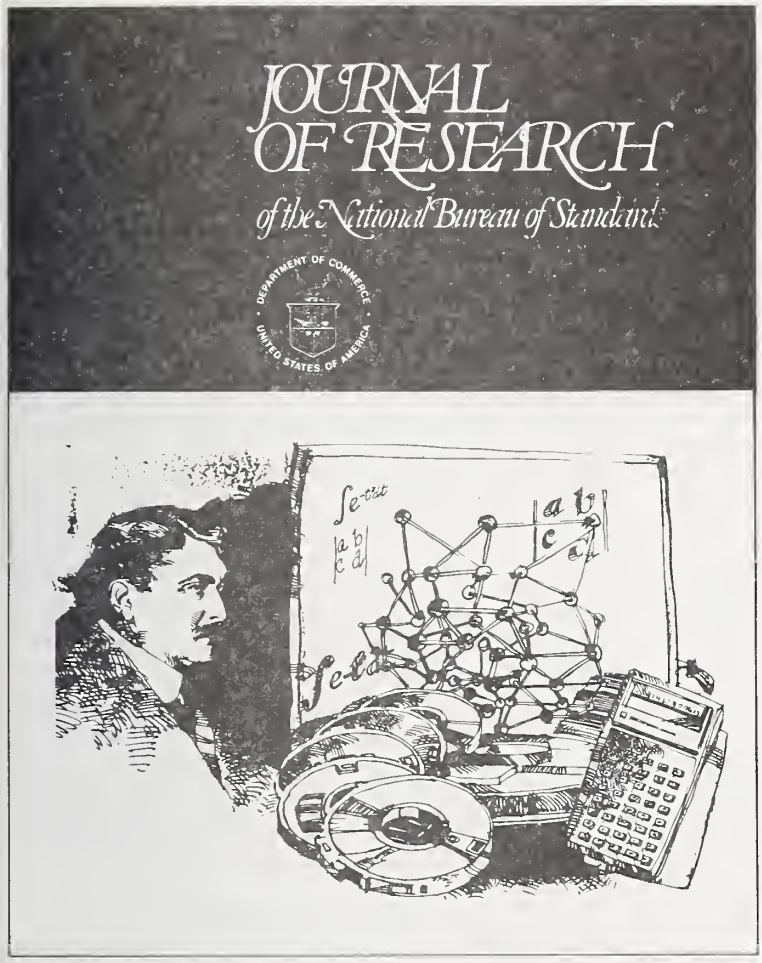
Name _____

Company _____

Address _____

City _____ State _____ Zip Code _____

(Notification key N-503)



Subscribe now — The new National Bureau of Standards Journal

The expanded 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 NBS publications in NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $17.00; foreign $21.25. Single copy, $3.00 domestic; $3.75 foreign.

- Note: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

For a review copy, write Journal of
Research, National Bureau of Standards,
U.S. DEPARTMENT OF COMMERCE
Washington, D.C. 20234

Subscription Order Form
Enter my subscription to NBS Journal of Research at $17.00. Add $4.25 for foreign mailing. No additional postage is required for mailing within the United States or its possessions.
(SJR—File Code 2Q)

Send Subscription to:

Name-First, Last

Company Name or Additional Address Line

Street Address

City | State | Zip Code

☐ Remittance Enclosed (Make checks payable to Superintendent of Documents)

☐ Charge to my Deposit Account No.

MAIL ORDER FORM TO:
Superintendent of Documents
Government Printing Office
Washington, D.C. 20402

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 NBS publications in NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $17.00; foreign $21.25. Single copy, $3.00 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, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the 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, $11.00; 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 world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396).

NOTE: At present the principal publication outlet for these 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. N.W., Wash., D.C. 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 purpose of the standards is to establish nationally recognized requirements for products, and to 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 **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.*

Order **following** *NBS publications—NBSIR's and FIPS 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. 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, $25.00; Foreign, $30.00.

Liquified Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $30.00. Send subscription orders and remittances for the preceding bibliographic services to National Bureau of Standards, Cryogenic Data Center (275.02) Boulder, Colorado 80302.

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

U.S.MAIL

SPECIAL FOURTH-CLASS RATE
BOOK