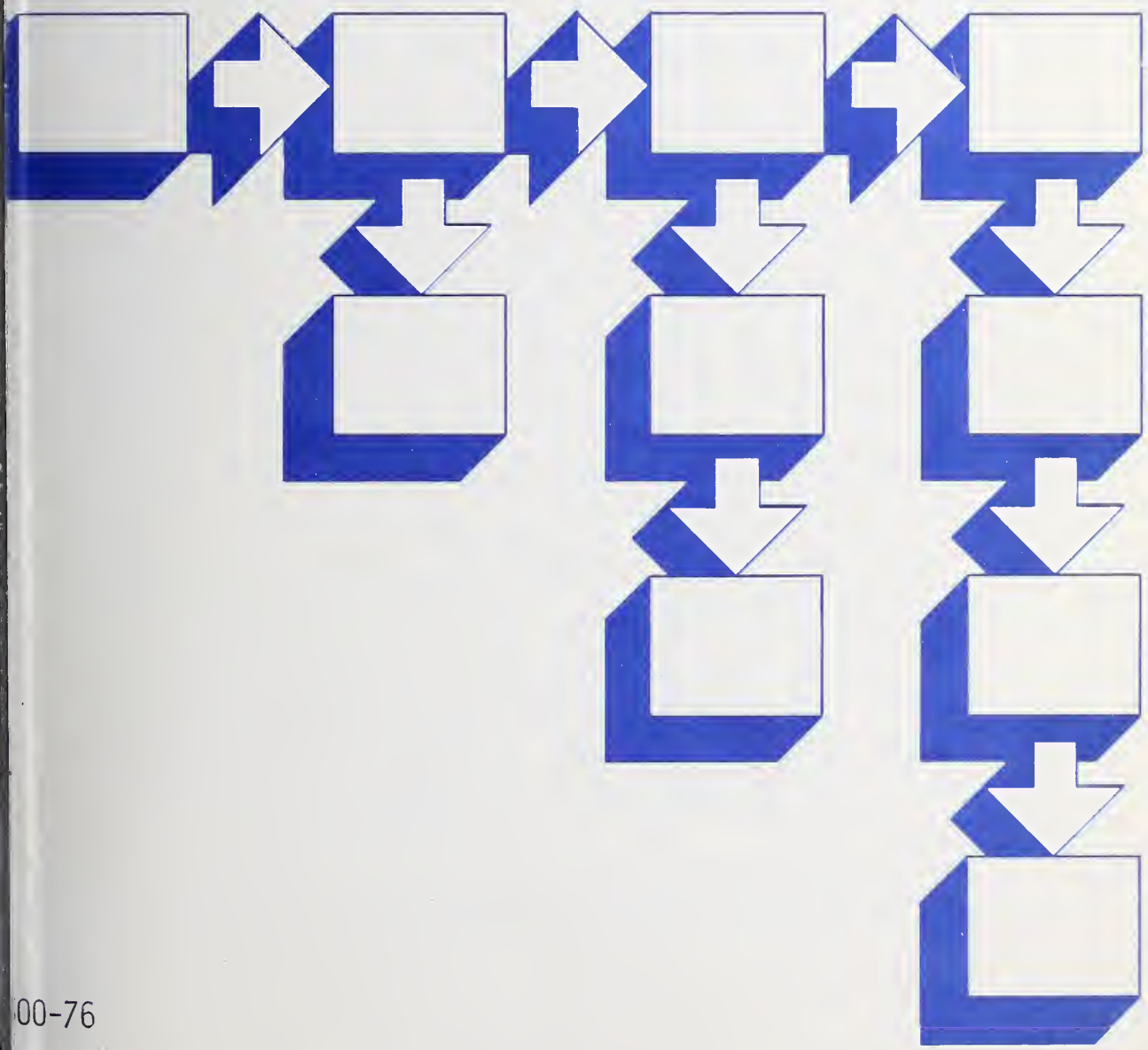
. Department
Commerce

onal Bureau
tandards

Computer Science and Technology

NBS Special Publication 500-76

Database Architectures— A Feasibility Workshop Report



00-76

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.

Computer Science and Technology

NBS Special Publication 500-76

Database Architectures— A Feasibility Workshop Report

Edited by:

John L. Berg

Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, DC 20234

Marc Graham

445 Euclid Avenue
Toronto, Ontario
Canada M6G2T1

Kevin Whitney

A. D. Little, Inc.
Acorn Park
Cambridge, MA 02140

U.S. DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Bureau of Standards
Ernest Ambler, Director

Issued April 1981

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 500-76
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-76, 64 pages (Apr. 1981)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 81-600004

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1981

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Price $4.00
(Add 25 percent for other than U.S. mailing)

FOREWORD

This report constitutes the results of two workshops held at the National Bureau of Standards to survey and report on the major technical consequences of implementing a three schema DBMS architecture, including the various implementation options and the identification of any necessary research.

Workshop 1, held on August 1-2, 1978, investigated the general topic of data independence. The participants were Mr. Charles Bachman, Dr. Thomas DeLutis, Dr. Rob Gerritsen, Dr. Eugene Lowenthal, Mr. Frank Manola, Dr. Alan Merten, Mr. Philip Shaw, Dr. Diane Smith, and Dr. Gary Sockut.

Workshop 2, held on August 22-23, 1978, examined supporting topics such as query languages, data dictionaries and database conversion. The participants were Dr. Donald Chamberlin, Dr. Eric Clemons, Ms. Nancy Goguen, Dr. Henry Lefkovits, Mr. David Shipman, Dr. Stanley Su, Major Anthony J. Winkler, and Dr. Carlo Zaniolo.

NBS gratefully acknowledges the assistance of the following individuals: Dr. Mani Daya contributed many ideas during pre-meeting planning sessions with the editors. Dr. John Smith and Dr. Gary Sockut offered comments which were included in the report. Dr. Alan Goldfine, NBS, played an important role in preparing the report for publication.

TABLE OF CONTENTS

DATABASE ARCHITECTURES: A FEASIBILITY WORKSHOP REPORT

John L. Berg,
Marc Graham,
Kevin Whitney,
Editors

To help the decision maker evaluate the potential benefits and pitfalls in moving forward with database technology, the National Bureau of Standards organized two workshops whose results are presented in this report. The workshops, held in August 1978, explored the progress plan and potential pitfalls involved in specifying, designing, and implementing systems based on the ANSI/X3/SPARC framework and the CODASYL JOD languages specification. Workshop 1 investigated the general topic of data independence, and Workshop 2 examined supporting topics such as query languages, data dictionaries, and database conversion.

Key words: Conversion; Database; Data-description; Data-dictionary; Data-directory; Data-manipulation; DBMS; Languages; Query; Standards.

1. INTRODUCTION

1.1 WORKSHOP OBJECTIVES

The workshops sought to explore the technical feasibility of a DBMS architecture with a high degree of data independence using the ANSI/X3/SPARC framework. The CODASYL 78 JOD languages are used, where possible, to insure specificity in the discussions. Workshop 1 was charged with:

1. Specifying criteria for determining the degree of data independence in any DBMS architecture in order to determine the degree of independence in the subject architecture.

2. Providing the criteria for dividing the functional components of a DBMS into internal, conceptual, and external schemas, and the mappings between any pair.

3. Identifying problems in the ANSI/SPARC approach to data independence.

4. Assigning the CODASYL specification statement classes to the schemas of the ANSI/X3/SPARC framework, using the criteria for data independence developed above.

Workshop 2 was charged with developing answers to the following questions in order to facilitate any eventual implementation:

1. How can we protect the user's investment in existing databases and application programs?

2. What is the role of the data dictionary/directory in preserving the user's data base investment?

3. Is there anything in this architecture which mediates against, or supports, end user use of the database? What specifications for end user facilities should be produced?

4. Is this architecture suited to the coming generations of database environments: distributed databases, special purpose machines, associative storage, advances in programming methodology, and other predictable technological advances?

1.2 MOTIVATION

1.2.1 Goals.

To help the decision maker evaluate the potential benefits and pitfalls in moving forward at this time with database technology, the National Bureau of Standards organized two workshops whose results are presented in this report. These two technology feasibility workshops of two days each brought together 16 industry and academic experts to explore the progress plan and potential pitfalls involved in specifying, designing, and implementing database management system technology based on the ANSI/X3/SPARC framework and the CODASYL JOD languages specifications. The first workshop was charged with discussing the degree of data independence provided by such an approach and the acceptability of the CODASYL languages specifications as candidates for the conceptual schema and external schema of the framework. The second workshop concentrated on implementation-related issues such as the role of the data dictionary, the supporting machine environment, and the distributed databases. The results of these workshops are recorded in this report.

1.2.2 Accomplishments.

The main product of the workshops is this report. It was edited from the notes, wall charts, and transcribed proceedings of the workshops. Whenever the editors felt there was substantial concurrence on a topic, they included it in the report. Where there were significant matters of particular interest to an individual participant in the workshops, that person wrote a position paragraph which is included in this report and attributed directly to its author. This is indicated in the text by the participant's name, which appears underlined and bracketed, immediately following the position. All referenced documents are indicated by an abbreviated name enclosed in brackets. The complete list of references can be found in Appendix A. The goal of the editors of this report was mainly to provide an outline in which these position paragraphs would fit.

If we have asked the right questions, provided a framework for analysis, and stimulated productive discourse on the technical feasibility of the ANSI framework for DBMS with these workshops and this report, our goals will have been met.

2. DATA INDEPENDENCE

2.1 INTRODUCTION

In his book on database management systems, C. J. Date emphasizes the importance of data independence by devoting a separate section to the concept. He writes that an application is data-dependent when

> ... knowledge of the data organization and access technique is built into the application logic. ... In a database system, however, it would be extremely undesirable to allow applications to be data-dependent. There are two major reasons: (1) Different applications will need different views of the same data ... (2) The DBA must have the freedom to change the storage structure or access strategy (or both) in response to changing requirements, without the necessity of modifying existing applications. [DATE]

While the goal of data independence is to isolate changes in the database from changes in the application view, a more precise definition was needed to permit the workshop participants to evaluate the degree of data independence provided by the ANSI/X3/SPARC framework.

A few published definitions will show the great diversity of views on exactly what data independence means:

> Data Independence has very specific properties, and can provide very specific predictable benefits ... even though data independence is so complex a phenomenon that it approaches confusion, it is possible to specify data independence functions and capabilities. [ANS75]

> Data Independence is concerned with the problems of separating application programs from some aspects of the storage and structure in the data base ... for the protection of investment in data and programs in a changing business and computing environment. Thus, "how much independence" is an economic question involving tradeoffs between flexibility and efficiency. [JARD]

> Data Independence is " ... the immunity of

applications to change in storage structure and access strategy." [DATE]

Stonebraker attempted to introduce a rigorous definition of data independence and to classify database transformations by the degree of independence provided, but little has been done with his model since. The issue to be discussed, then, is can we define data independence in a precise and measurable way and, if so, how much do the ANSI architecture and CODASYL provide?

Before starting into discussions, the workshop participants prepared the following list of key terms for which standard definitions would facilitate their work:

Data Independence	Schema
Data	Conceptual
Information	Internal
Integrity	External
Constraint	Logical data structure
Index	Mapping

Of these items, the workshop specified data independence, the (three types of) schemas and mappings as most important for their discussions.

2.2 DATA INDEPENDENCE DEFINED

The workshop discussion began with the workshop leader's prototypical definition:

> Data independence protects (a user's) investment in databases and programs by insulating the user from inevitable changes in applications, data, and computer systems.

Although this is a statement of objectives rather than a proper definition, the workshop participants improved it to:

> The objective of data independence is to permit the use of information in a changing environment. [Smith]

This point was later expanded in a joint position paper by Diane and John Smith:

The term "data independence" has different, though related, meanings when applied at the CONCEPTUAL level and the EXTERNAL level.

i. Meaning at the CONCEPTUAL level: Data independence is achieved when the relevant structure of the enterprise is revealed, and its representation over computer storage is hidden. It is not sufficient merely to hide representation details--this would imply that an "empty" conceptual schema would be an adequate solution.

ii. Meaning at the EXTERNAL level: Data independence is achieved when the relevant structure of the enterprise appears to the user in a desirable external representation, and the internal representation over computer storage is hidden. It is possible that the internal representation will be chosen to reflect the desired external representation. [Smiths]

Data Independence is the property of a data management system that provides alternate views of the same stored data, and preserves them during the evolution of the data environment. [Manola]

The objective of data independence is to permit the continued acquisition, storage, retrieval, and dissemination of information in support of the operation of the enterprise over time. The effect is to insulate the enterprise from the inevitable changes that occur in applications, data, computers, and the enterprise's view of itself. [DeLutis]

These attempts to agree on an intensional definition (one which designates the qualities of objects to which it applies) having failed, the group turned to devising an extensional definition (one which designates the objects to which it applies) of data independence. The workshop participants devised this extensional definition of data independence by enumerating the classes of changes permitted by a database system with a high degree of data independence. The following list of capabilities as relating to data independence was recorded and augmented by Bachman:

a. changing character of floating point number or integer representation,
b. changing record delimiter mechanism,
c. changing names of records, sets, and items,
d. changing location within record of item,
e. adding/subtracting unreferenced items,
f. changing internal format of items, including coding schema,
g. changing precision of data items to make more precise,
h. changing units of measure,
i. adding new record and set types,
j. adding consistency and derivation declarations,
k. changing data model,
l. refining conceptual schema declarations,
m. changing primary and secondary key indexing techniques,
n. changing set implementation technique,
o. changing location of records from one site to another,
p. changing number of records or specific content which represent a single real world entity.

Throughout this discussion it was clear that various levels of data independence were possible--corresponding to the characteristics of the data that could be changed between storage and use. Each schema is a description of the data, with emphasis on various characteristics of the data as shown in the following table:

ANSI schema	Data characteristic	DIAM level
external	format	end user
conceptual	structure	infological
	access	string
internal		encoding
	representation	physical device

There may be mappings between each level, with a degree of data independence permitted at each level. Some changes such as renaming a field may occur at one or more levels, while others such as selecting the indexing mode occur only at one level.

Several participants used a model like this one to clarify their view of some aspects of data independence. Gerritsen argued, for example, in the following position paragraph, that machine independence is not a concern of DBMS data independence:

Computer independence should be considered separately from the overall question of data independence because:

1. Change in computer and/or DBMS occurs much less frequently than changes in data, applications, and enterprise model.

2. Insulating programs from changes in computer transcends the capability of the DBMS. Examples of changes from computer to computer that affect programs but cannot be controlled by the DBMS include:
   a) collating sequence
   b) number of distinct characters
   c) number of characters stored per word
   d) precision of arithmetic.

3. A large degree of computer independence can be attained by means that have no relation to the means used to obtain independence from changes in applications, data, and the enterprise model. Computer independence can be attained simply by implementation of the DBMS on a variety of computers. Examples of DBMS that are currently available on a large variety of computers are TOTAL and SEED. [Gerritsen]

Using examples from his paper on database reorganization, Sockut divided database changes into four DIAM levels as follows:

Changes at the infological level (conceptual schema):

- attributes can be added, deleted, combined, split, or renamed

- relationships can be created, destroyed, or renamed

- migrating an attribute in a 1-N relationship from the 1 to the N or vice versa

- changing among 1-1, 1-N, and M-N relationships.

Changes at the string level (internal schema):

- creating, destroying, or renaming a string (record or set)

- rearranging fields on a record or records in a set

- establishing or removing a secondary index, or a search key within a record type that is densely indexed on that key

- changing the order of a set's members.

Changes at the encoding level (internal schema):

- modifying the basic representation, scale, encryption, size, precision, character code, etc., of an attribute encoding

- modifying the relationship encoding as, for example, changing set implementation among embedded chains, pointer arrays, and bit maps.

Changes at the physical device level (internal schema):

- changing access methods

- changing hash parameters

- remapping areas to devices

- eliminating overflow. [Sockut]

The focus on levels of data description and the mappings between them clearly delineates the major distinction between a two schema and a three schema DBMS framework. In a system using only two schemas, one for representing data as stored and another for representing data as used, there will be N x M mappings for N storage views and M application views. When a storage view changes, M mappings must change; when an application view changes, N mappings must change. By contrast, a three schema system has a single conceptual view separating the storage views from the application views. In this case, only one mapping changes if a storage view or an application view changes.

At run time these two system organizations produce the same database manipulations, so they provide an identical degree of data independence. Thus, the important question is not the theoretical degree of data independence (complexity of mappings from storage to usage), but the practicality of providing that degree of data independence. A capability which is so inconvenient that it is never used provides no greater data independence than one which is theoretically nonexistent. In this sense, the three schema approach provides much more data independence, since a change requires changing only a single mapping rather than N storage or M usage mappings.

This distinction between theoretical and actual data independence is clearly expressed in Lowenthal's definition:

> Data Independence is a property of the program interface provided by a DMS; namely, it is a (necessarily qualitative) measure of the extent to which changes in the implementation of the interface, and specifically changes in the organization of the data, can be made without the requirement to modify programs.
>
> Potential data independence more precisely refers to the changes which could be made. This can be determined from the interface itself.
>
> Actual data independence refers to those aspects of the potential data independence which are actually supported by the DMS--i.e., the existence of mechanisms to 1) change the data and 2) continue to preserve the program's view of the data.
>
> As with other measures of user interfaces, such as "high-levelness," "ease-of-use," etc., data independence is not absolute, but must be related to the needs of the enterprise. [Lowenthal]

2.3 THE THREE SCHEMA FRAMEWORK

2.3.1 ANSI Schemas and Their Mappings.

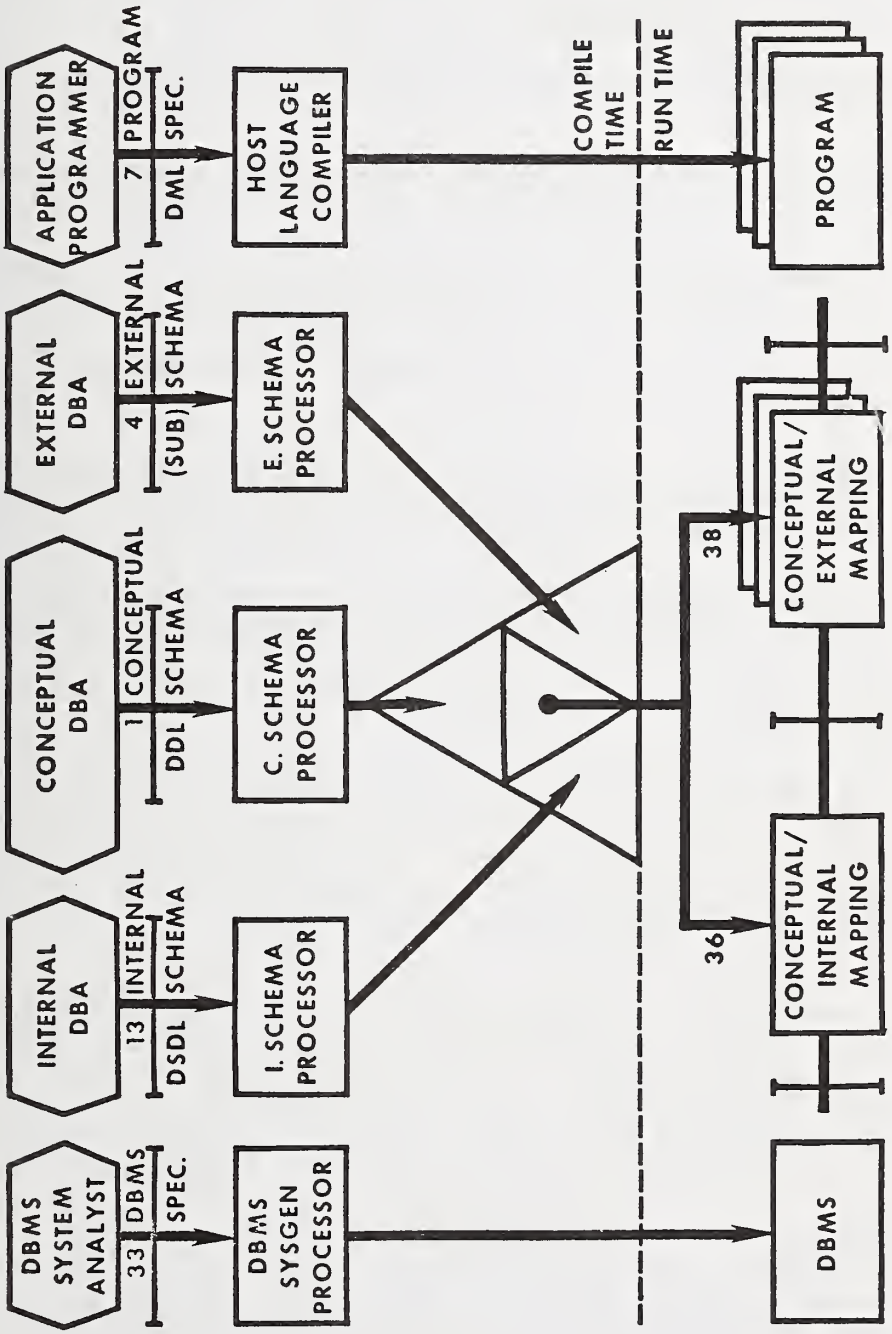
After discussing the aspects of data independence, the workshop proceeded to investigate the ANSI framework and specifications for its schemas and the mappings between them. These schema definitions and mappings provide the measure of data independence being sought.

Figure 1 shows the subset of the ANSI/X3/SPARC architecture of primary concern to our discussion. Only those interfaces of direct interest to the data independence workshop are shown. These are:

1) Conceptual Data Description
4) External Data Description
7) External Data Manipulation Language
13) Internal Data Description
36) Internal/Conceptual Data Transformation
38) Conceptual/External Data Transformation

The internal, conceptual, and external schemas are defined in the ANSI/X3/SPARC documents [ANS75], [ANS77]. In general terms, the conceptual schema represents a long-term view of the enterprise independent of its databases or information processing applications. The internal schema is a description of the organization's databases. An external schema is a description of the data used by a collection of application programs of the enterprise. Logical data elements of an external schema need not correspond in a one-to-one way with physical data fields described in the internal schema. The values of external data items may be translations, transformations, concatenations, logical or arithmetic computations, or other algorithms performed on one or more internal data values.

Figure 1
ANSI/SPARC DBMS Study Group Architecture



2.3.2 The Conceptual Schema.

Some criteria suggested by workshop participants for permitting a language element to be in the conceptual schema or to be excluded from it were the following:

- The function declared by the language element should be appropriate for the schema
- This type of language element is required for the schema
- Each function should be described in only one schema or part of a schema
- There should be only one way of representing relationships in the conceptual schema
- Descriptions in the schemas should not duplicate one another
- Nothing in the conceptual schema is ignorable; each declaration must have effect or import to the enterprise administrator
- The conceptual schema factors repetitive declarations out of the internal and external schemas.

Smith commented on what should be specified at the conceptual level:

> The conceptual schema is intended to capture all aspects of an enterprise necessary to provide support information for its operation. Thus, both the structural and behavioral aspects of the enterprise must be specified.
>
> Structural aspects consist of the entities, attributes, categories, and relationships comprising the enterprise as well as the naming mechanisms used to reference them. Important naming mechanisms are: associative naming (e.g., the employee with ID# = 99999); operative naming (e.g., the employee last hired); functional naming (e.g., the number "2+3"); and relationship naming (e.g., the employee who is assigned to project C). It is of utmost importance to note that it is unlikely a given structural component is interpreted in the same way throughout the enterprise. For example, in a hotel the object "reservation" may be interpreted by some class of users as an entity with attributes "date,"

"reservee," and "room." A second class of users may interpret "reservation" as a relationship among the entities "reservee," "date," and "room," where "reservee" has attributes "name," "address," etc. Another class of users may view a "hotel reservation" as one subcategory of "reservation" among many such as "plane reservation," "library book reservation," etc. Thus, the conceptual schema must specify the structure in a way that permits all these different interpretations to be captured.

The behavioral aspects of an enterprise consist of the operations performed within an enterprise. Such operations are reflected in the information system as insertion, deletion, and update operations applied to its structural aspects. It is important to note that the effects of such operations are rarely restricted to isolated objects in the structure. Rather, they have side effects that ripple over related objects. For example, in a personnel application the termination of an employee would require not only his deletion from the system, but also his removal from health plans, car pools, and project assignments. It would probably also require new assignments being made to fill some of these newly emptied slots. These side effects must also be captured in the conceptual schema--either as an aspect of an operation or as an integrity constraint specified separately from the operation. [Smith]

2.3.3 The External Schemas.

Workshop participants seemed generally in agreement on the nature of external schemas, making such comments as:

- The CODASYL subschema is a proper example of an external schema
- The external schemas will be allowed to use data models convenient to the user's applications and programming languages
- Multiple concurrent external schemas are needed to isolate one user's view from another user°s view of the database. For example, record order and structure should be variable from one external schema to another

- The external schema is the proper level of granularity for access control.

2.3.4 The Internal Schema.

The workshop also discussed the internal schema briefly, with comments such as:

The CODASYL DDL and DSDL form a proper example of a possible internal schema:

- When a set expresses the binding of a relationship, it is properly part of an internal schema

- Similarly, when a set expresses an access path, it belongs in the internal schema.

Only one position paragraph was written on the internal schema, arguing that its existence need not be known by the users of a data management system or by the enterprise administrator:

> Given a conceptual schema, a set of external schemas, data volumes, access frequencies, logical access path frequencies, storage device characteristics, performance requirements, and other performance/storage constraints/costs, then it is possible to represent internal schema design decisions in a mathematical optimization model. Such a model can perhaps be solved "optimally" or, through heuristics, be used to find "good" solutions. Such a model can be incorporated in the DBMS. Since the DBMS only creates, modifies, and uses the internal schema, the user need not know of its existence. Hence, except for internal requirements of the DBMS, the concept of "internal schema" can be estimated. [Gerritsen]

2.4 USING DDLC JOD78 IN THE ANSI FRAMEWORK

Rather than allocating DDLC language elements to the three schemas, the workshop participants grouped all data description language elements into the conceptual schema. Then, those relating to the storage and efficient retrieval of the data were moved to the internal schema, and those relating to user views, applications, and programming languages were moved to the external schemas.

Smith made the following statement about the inclusion of operations in the conceptual schema:

> The conceptual schema should describe more than just the structure of an enterprise. It is equally important to describe its behavior as characterized by the operations that effect changes within it. Such operations can be defined over the components of the structure without introducing any implementation detail. Each operation should be specified in terms of basic insert, delete, and update operators, high-level naming mechanisms, and well-conceived control structures. The use of appropriate naming mechanisms makes it possible to refer to objects without considering access paths and other representation detail. Control structures are necessary to define the scope of the objects affected and could take the form of quantifiers or more conventional constructs such as recursion and iteration.
>
> Such an extended specification provides several advantages:
>
> i. it permits users to understand how change is effected in the enterprise,
>
> ii. it provides implementors with a basis for implementation optimization,
>
> iii. it provides DBAs with a basis for verifying that high-level integrity constraints will be maintained,
>
> iv. it provides a basis for access control--if all accesses are channeled through only the specified operators.

References:

1. Liskov, B., and Zilles, S. "Programming With Abstract Data Types." Proc. of a Symposium on Very High Level Languages, SIGPLAN Notices 9, 4, April 1974.

2. Mylopoulos, J., Bernstein, P., and Wong, H. K. T. "A Language Facility for Designing Interactive Database-Intensive Applications." Supplement to Proc. 1978 ACM SIGMOD Conference, Austin, Texas, May 1978, 15-25.

3. Smith, J. M., and Smith, D. C. P. "Integrated Specifications for Abstract Systems." University of Utah, Computer Science Dept. Report UUCC-77-112 (Sept. 1977). To appear in IEEE Trans. on Software Engineering. [Smith]

This discussion of which data description language statements should belong in the conceptual schema was organized according to the DDLC's nine categories of language elements. A summary of the allocation of language elements is shown in Table 1, which should be interpreted according to the following two paragraphs:

The schema category identifies a schema and names its characteristics. Its syntactic elements include the schema access-control clause, the schema CALL clause, and the SCHEMA NAME clause. The schema category inspired no comment.

The structure category names the data structures that the schema describes. The syntactic elements of this category include the Data-name clause, the KEY clause, the OWNER and MEMBER clauses, the SET-NAME and RECORD NAME clauses, the OCCURS clause, and the ORDER clause.

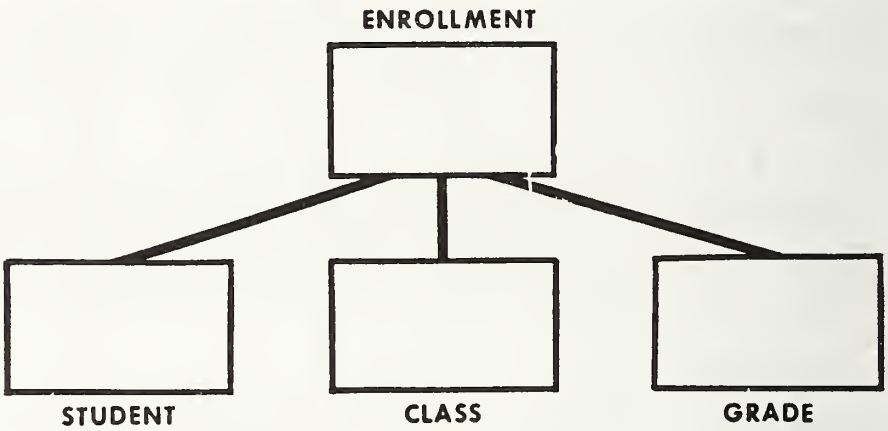
Discussion of the structure category focused on the adequacy of the CODASYL set for defining the relationships among entities described by the conceptual schema. Several participants objected to the multiple uses of the CODASYL set construct, as explained in the following position paragraphs. Smith objects to the use of SET to show access path information.

Table 1
DDLC Allocation of Language Elements

LANGUAGE ELEMENT CATEGORY	ASPECTS INCLUDED IN SCHEMAS		
	INTERNAL	CONCEPTUAL	EXTERNAL
1. SCHEMA	X	X	X
NAME, CALL, ACCESS CONTROL			
2. STRUCTURE			
NAMES	X	X	X
KEYS	PERFORMANCE		ACCESS
SETS, GROUPS	ACCESS PATHS	RELATIONSHIPS	
ORDER	PERFORMANCE	SEMANTIC	PRESENTATION
MAPPING			
PLACEMENT			
REPRESENTATION			
3. VALIDATION			
CHECK, PICTURE, TYPE	REPRESENTATION	DOMAIN	PRESENTATION
IDENTIFIES, INSERTION		X	
CONSTRAINTS, SOS		X	
SOURCE/RESULT		X	
4. DML INTERFACE			X
5. ACCESS CONTROL		X	X
6. MEASUREMENT	—	—	—
7. TUNING	X	X	X
8. RESOURCE ALLOCATION		X	
9. ADMINISTRATION	X	X	X

Certainly the SET and RECORD statements can be used at the conceptual level to specify relationships between objects. However, in practice, these statements are frequently used to specify the access paths that are to support the relationships. For example, consider the following schema and three CODASYL descriptions:

Figure 2
Enrollment Example




```
A. RECORD      Enrollment   /* Flat File Model */
     05          Name
     05          Class #
     05          Grade #

B. RECORD      Enrollment   /* CODASYL Model */
     05          Grade #
   RECORD      Student
     05          Name
   RECORD      Class
     05          Class #
   SET         SE
     OWNER       Student
     MEMBER      Enrollment
   SET         CE
     OWNER       Class
     MEMBER      Enrollment

C. RECORD      Enrollment   /* Relational Model */
     05          Name
```

```
    05          Class #
    05          Grade #
  RECORD      Student
    05          Name
  RECORD      Class
    05          Class #
  RECORD      Grade
    05          Grade #
```

This structure implies a large number of access paths, many of which are redundant.

Conceptually, each of the CODASYL descriptions permits the same information to be extracted--but using different access paths. Thus, RECORD and SET are good constructs for specifying access paths--an internal schema process. This freedom to specify the enterprise in multiple ways, employed at the conceptual level, would yield multiple specifications of the enterprise. This requires a schema reconciliation step that could be avoided either by disciplining the use of constructs or by providing a single construct that does only what is wanted. The second option is usually more practical to implement. [Smith]

Sockut amplifies this point, and also has the same complaint about the OCCURS clause:

The CODASYL Set is inadequate as a mechanism for describing relationships in the conceptual schema because:

1. The only way to specify a relationship is to define a SET or OCCURS, which unfortunately specify the relationship's implementation as well. It would be better to separate relationship definition (C-level) from relationship implementation definition (I-level and possible E-level as well).

2. SET sometimes means both a relationship and an access path (if it is a non-singular set), and sometimes it means only the access path (if it is a singular set).

Here is my view of the ideal:

1. Conceptual schema level: Specify relationship including its multiplicity (i.e., 1-1, 1-many or many-many).

2. Internal schema level (and possibly external schema level): specify implementation of relationship (e.g., SET vs. OCCURS). Also specify access paths which do not implement relationships (e.g., a SET with OWNER=SYSTEM). [Sockut]

Lowenthal also comments:

> CODASYL DDL provides two ways of defining 1-N relationships--sets and nested repeating groups. At the conceptual level, there should be only one way of expressing a relationship; linked list (set) vs. contiguity (repeating group) should be an internal consideration. [Lowenthal]

Since the OCCURS clause affects the data storage structure, it should be moved to the internal schema from the conceptual. If an external schema is used with a programming language which supports repeating groups, then OCCURS may also be used there.

> The OCCURS clause affects both the internal and external schemas. There seems to be agreement that OCCURS affects the external. It affects the internal because it controls implementation of a relationship through physical contiguity rather than with pointers. The presence of the OCCURS in the DDL as well as in the COBOL subschema DDL is a clear indication that CODASYL intends for the presence of OCCURS to affect the storage structure. [Gerritsen]

There was considerable discussion of the ORDER clause and its roles. Performance aspects belong in the internal schema, semantic aspects belong in the conceptual schema, and presentation or display aspects belong in the external schemas. The following comment argues for excluding ORDER from the conceptual schema:

> The effect of supporting ORDER as a reasonable conceptual schema attribute has a negative effect on data independence (in the CODASYL case) because a given set type can have only one ORDER clause in the CODASYL DDL. A set may have, in the real world, several valid orderings--i.e.,

different users perceive different orderings.

> If the enterprise administrator changes his mind about which mechanism is to be used to express an ordering, some programs will have to be changed. Contrast this with the relational approach where all orderings are item based and all are symmetric. [Lowenthal]

Smith, however, notes an important exception to this point:

> When order is used to improve the performance of a database management system (DBMS), then it should be considered an implementation mechanism and specified in the internal schema. However, when an ordering is used within an enterprise (to facilitate either communication or its activities), this should be reflected in the conceptual schema. Typically queue and stack disciplines may be used to service customers. If the service operators are included in the conceptual schema (see position paper by D. Smith on the inclusion of operators in the conceptual schema), then the notion of order is necessary to utilize these operators as intended. This capability extends the applicability of DBMSs beyond their current use as record keeping systems to the more dynamic situations inherent in control applications. [Smith]

Another DDLC structure clause which caused debate was the KEY clause which defines entry points to the database. The primary objection is expressed in this position paragraph:

> The "KEY - IS" clause in CODASYL 1978 detracts from data independence objectives in that the intent of the clause is to define an "entry" to the record type from the DML. The effect is to bind the DML to the DDL. E.g., if I modify the schema to delete the record key attribute from an item, then all programs which used the item as an entry point will now be invalid--the DML processor will reject the FIND. [Lowenthal]

The validation category declares rules that constrain occurrences of the data structures declared in the structure category. The syntactic elements of this category include the CHECK and DUPLICATES clauses, the record IDENTIFIER clause, the TYPE and PICTURE clauses.

Data validation was generally agreed to be a function of the conceptual data description except for the following statements and clauses:

- The CHECK clause specifying range restrictions of a data element may also appear in an external schema as long as the constraint in the external schema is more restrictive than the constraint in the conceptual schema.

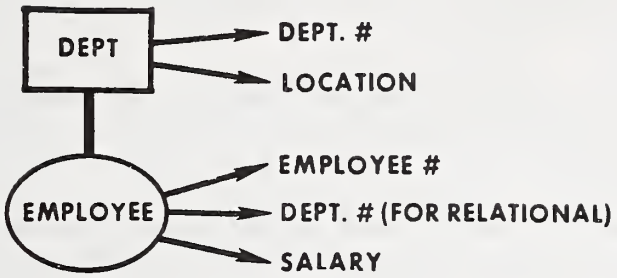
- Those aspects of the PICTURE and TYPE clauses that express the precision or domain of data elements belong in the conceptual schema since they specify semantic information about the data items in the database. Those aspects which express the format and presentation of the data items belong in the external schemas.

- The structural constraint clause is a way of specifying a relationship rather than a data value constraint. In this sense, it more properly belongs in the structural category than in the validation category.

The DML interface category declares procedures which may be involved by a DML function and parameters to be supplied in these procedures. The syntactic elements of this category include the RANGE KEY clause, the set occurrence SELECTION clause, and the WITHIN area clause. This was the only time in the discussion that the interrelationships of the DDL and the DML were explicitly addressed. Lowenthal commented in two position paragraphs:

> We have ignored the effect the DML has on data independence characteristics--remember my definition of data independence refers to the "program interface" which includes the DML. For example, consider the N-1 relationship:

Figure 3
An N-1 Relationship



The query expressed in S2000 query language is:

```
WHERE salary GT $10,000 AND
      location EQ plantQ
```

and the same query expressed in a relational calculus language is:

```
WHERE dept.location EQ plantQ
   AND THERE EXISTS  employee :
      dept.dept# EQ employee.dept#
   AND  salary  GT  $10,000
```

Note that the S2000 statement refers only to items (domains) and not records (relations). Therefore, the statement is independent of whether LOCATION is in the department record or in the employee record. This is not true in the relational DML example--it is assumed that LOCATION is in DEPT. If LOCATION moves to EMPLOYEE then either the statement is invalid or else there must be a mapping which allows the external view to see LOCATION in DEPT--I'm not sure this is definable. In other words, a DML which minimizes references to record names supports item migration as an aspect of data independence. (Of course, the CODASYL DML also refers explicitly to record names). [Lowenthal]

The access control category declares authorization mechanisms for access to and change to the occurrences of the data structures declared in structure category. The syntactic elements of this category include the ACCESS-

CONTROL clauses.

Time did not permit the workshop participants to explore this subject fully. The following points were made and Manola contributed a more detailed position paragraph:

- Access control is actually a property of the mappings between schemas, rather than the schemas themselves.

- Access control does not affect the actual data independence, but may have an effect on the practical data independence of a data management system.

- The access control information belongs partly in the conceptual and partly in the external schemas.

- The proper mechanism for access control at the external level is the external schema (CODASYL subschema). That is, each distinct set of user's access rights to the database should be defined in a separate subschema. Particular subschema have identical access rights designated for that subschema.

Manola comments:

> Access control functions go into the conceptual schema (as well as in other schemas) for the following reasons:
>
> 1. Access control rules (like integrity rules) are part of the description of the enterprise the conceptual schema is supposed to model. Generally speaking, no one objects to the idea that customers, suppliers, and products are entities of interest to certain types of conceptual schemas, and should be defined in them. Moreover, few people object to the idea that, if there are rules of various kinds about which suppliers supply which products, or which products are shipped to which customers, these sorts of rules legitimately belong in the conceptual schema (I can imagine a DOD rule that says we don't ship weapon x to country y). Certainly products of various kinds (as well as plants, warehouses, etc.) are enterprise resources that we need to keep track of,

and may well be described in the conceptual schema. Well, the database of an enterprise is an enterprise resource which we need to keep track of (or so database people have been preaching for years), and which could well be described in the conceptual schema. Moreover, users of that database system could also be viewed as entities of interest, and the semantically meaningful relationships between these users and the database (plus the definition of the properties of legitimate users itself) describe in some sense the security rules of the system. That the ANSI/SPARC framework envisaged this approach is shown by page vi of its report where it says "By defining the persons with access to the database management system as entities of interest, it is possible to directly model the rules of access and thus provide the necessary access control at the level of the conceptual schema. Access may be further limited at other levels."

2. I would initially imagine security policies being defined at the conceptual level. This would include overall government regulations under which the company must operate that are relevant. For example, a privacy-type policy might be one which states that any person can see the data stored about himself in the database. Another policy might be the normal DOD security policy that a person might have the appropriate clearance before he can see a piece of classified data. The actual implementation of these policies might be specified at the internal level (e.g., how the system stores the fact that a piece of data has a certain classification, or that a user has a specific clearance).

3. I view the separation of the usual data in a database and the security and integrity rules, information about users which these rules might require, etc., as being a very artificial one. I see no reason why I should not be able to ask the DBMS what users have access to a

particular piece of data or document, just as I ask what suppliers supply part xyz. Having to go through an entirely independent mechanism to access this type of information seems both unwieldy and insecure. (There are a number of other types of "metadata" which I might want to get from the database too, such as what relationships do objects x and y have in common.)

4. It is inappropriate to leave access control to external schemas, let alone to the operating system. The operating system is able to control access in terms of the objects it knows about. Generally these are rather coarse objects like files. The database system will have to be responsible for enforcing access rules on those objects which it alone has defined and knows about (while the operating system protects the physical representations and the database management system itself). While use of the external schema facility to define "security views" is one way of implementing access control (and one which I advocate in many cases), it is not the only way. I would not like to see any assumptions made here unduly constrain users in how they implement their access control. Moreover, this appears to require the burying of the access control rules in the mapping specifications for the external schemas, rather than having them explicitly declared somewhere. [Manola]

The measurement category directs the DBMS in collecting data about database use, population, etc. Since there are no syntactic elements of the 1978 JOD DDL in this category, the workshops did not discuss this category.

The tuning category declares guidelines for database organization to assist in tuning database performance. The syntactic elements of this category include SOURCE and RESULT clauses, the PRIOR processable clause, the LOCATION mode is clause, the LINKED to clause, and the SEARCH clause.

There was considerable discussion of this class of language specification, but little consensus. The main points brought out by the discussion were:

- Tuning options affect practical data independence, but not theoretical data independence.

- It is important to distinguish volume of data from frequency of use in specifying tuning statements.

- Record usage statements should be in the conceptual or internal schema to reflect global data usage--rather than the data usage of a single class of applications related by an external schema. (See position paragraph following this list).

- Tuning should be an internal schema matter and useful only to the DBMS which might use the information to modify the schema automatically.

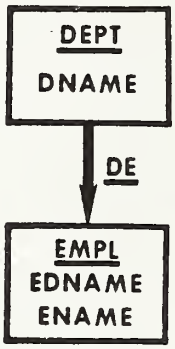
Three position paragraphs were submitted on this topic:

> From a formal point of view, a tuning declaration (such as frequencies of use and volumes of conceptual entities) exposed in terms of conceptual objects is part of the declaration of the internal-conceptual mapping (since it affects how the conceptual objects will be represented at the internal level). From the point of view of packaging such declarations into various languages, it will probably be convenient to package them in the conceptual schema, as this is a central point for collecting global declarations about the system as a whole. [Manola]
>
> Performance oriented estimates/projections should include frequency and volume data. They are optionally included in the conceptual schema expressed in terms of abstract observable properties of conceptual schema objects. They may be of use to:
>
> - internal schema designers
> - automatic internal schema generators
> - DBMS for automatic tuning of the database in ways which may not even be reflected in the internal schema.
>
> Note that some performance related information

could be provided in a program (i.e., in DML or external schema)--e.g., "I intend to retrieve 4000 employee records"--which will cause the DBMS to stage the data to high speed storage or take other appropriate action. [Lowenthal]

The SOURCE clause is appropriately categorized under the EXTERNAL schema, not under the CONCEPTUAL schema. It is an EXTERNAL schema component because it permits construction of a record that contains values derived from relationships. For example, consider

Figure 4
SOURCE clause example



EDNAME has as SOURCE DNAME. This permits a view (EXTERNAL schema) without the DEPT record and without the DE set in which an employee's dept name (as EDNAME) is still available. Contrary to assertions made at the meeting by Frank Manola and Charles Bachman, the SOURCE clause does not act as a clause associated with data base integrity (if it did it would properly be associated with the conceptual schema as well). The JOD specifically states that SET SELECTION must be used to accomplish this kind of integrity. SOURCE allows one to tell the DBMS--"replicate an item here and give its replication a new name." Since one could do this for all items in records that own sets (into the members) without change to the enterprise model, or to allowable data, it seems not to be associated with the conceptual schema. [Gerritsen]

The resource allocation category names organizational units appropriate for managing system resources, and controls the assignment of occurrences of the declared data structures to those units. The syntactic elements of this category include the AREA NAME clause and the WITHIN area clause.

The administration category names and provides for the invocation of DBA supplied procedures. The syntactic elements of this category include the CALL clause.

The workshop noted that database procedures, like all other forms of user exits, reduce portability. In that sense, they serve to reduce data independence.

2.5 CONCLUSIONS

The group agreed to the following basic conclusions:

- No intensional definition of data independence is possible.
- The two schema and the three schema architectures provide the same degree of theoretical data independence.
- The three schema architecture provides a greater degree of practical data independence.

3. IMPLEMENTATION ISSUES

3.1 INTRODUCTION

While the participants in the first workshop had been charged with the examination of a single issue, data independence, the second workshop was charged with four large areas. These areas relate very closely to implementation problems, in contrast to the more nearly theoretical concerns of the first workshop. These areas were not addressed in any detail by the ANSI/SPARC study group report, nor by the latest CODASYL Journals of Development. The main task of the second workshop was to find the pitfalls in the ANSI/SPARC framework and pitfalls in the CODASYL JOD specifications as an implementation of that framework in the context of these four areas.

Not all the participants agreed that the CODASYL specifications were, or could become, an implementation of the ANSI/SPARC framework. The following positions were submitted:

> Comparison of CODASYL and ANSI/SPARC Architectures. I take strong disagreement with the premise that CODASYL 78 represents a valid embodiment of the A/S three schema database architecture. While there are three schemas in CODASYL 78, there are not the required three schemas: it is not accurate to equate the DDL schema with the conceptual schema nor the subschema with external schema. The DDL schema is not quite the conceptual schema. Too much of the internal level remains, even with factoring out pointers and indices into the DSDL. And too much of what belongs at the conceptual level remains unspecified, or provided only through vaguely specified CALL LOCK, or BY PROCEDURE references. Most damaging is the inadequacy of the subschema facility provided by CODASYL 71, which the revisions of 78 did not address. This cannot be considered an ANSI/SPARC external schema facility: it restricts subschemas to resemble too closely DDL schema, thus leaving conceptual and internal constructs visible in the user interface. This is not a small objection, nor is it easily fixed; by closely linking subschema data access to DDL schema records and sets, serious limitations have been placed at

the level and power of available DML statements. This in turn has serious implications for programming effort and productivity, channel traffic in distributed processing, and security and integrity. [Clemons]

Lefkovits noted:

> The proper issue to be discussed here is not whether CODASYL 78 is an adequate representation of the ANSI/SPARC architecture, but rather whether CODASYL 78 sits into that architecture. In my opinion it is obvious that the answer to whether one can equate CODASYL 78 to ANSI/SPARC must be a categorical negative; for example, CODASYL 78 is a two schema structure and whereas there is nothing in CODASYL 78 that inhibits the introduction of a conceptual schema, this has not been done to date. On the other hand, there is nothing in CODASYL 78 which appears to be inconsistent with the ANSI/SPARC architecture and as such, CODASYL 78 does not inhibit any of the benefits that can be accrued from this architecture. [Lefkovits]

3.2 DATABASE ENVIRONMENT

The workshop discussed the impact of current hardware and software technology research on future database practice. The hardware developments discussed were database distribution and specialized database machines ("backends"). The software developments were structured programming, modular design, and abstract data types. [LISK], [GUTT].

3.2.1 Distributed Databases.

The workshop discussed two questions: one referencing ANSI/SPARC and one referencing CODASYL. The ANSI/SPARC framework was seen by some workshop participants as being well suited to the distributed database environment. Others thought that the ANSI/SPARC report implied a single coherent database. Database distribution was said to be a problem similar in nature to the earlier problem of interrelated data on disparate storage media and of differing storage structure.

Lefkovits made this comment on the incompleteness of the ANSI/SPARC architecture:

> The ANSI/SPARC architecture does not address itself to conventional file organization and distributed data (databases and/or files). These two subjects are similar in some ways as they both address data located outside a single database. There is nothing in the current document that contradicts such future extensions, other perhaps than the fact that the role of the data dictionary/directory will have to be augmented substantially to reflect these changes. In an identical manner CODASYL 78 does not address these subjects either, but it appears that the same extensions that could be made to the ANSI/SPARC architecture could be included in some future version of the CODASYL specifications. [Lefkovits]

The ANSI/SPARC conceptual schema represented to most of the workshop a good vehicle for coordinating separate data models and DBMS, but distribution itself was felt to have ramifications throughout the framework. It was not isolated in, for example, the internal schema or its mappings.

The workshop was in near agreement that, in its current state of development, the CODASYL specifications could not adapt well to a distributed database environment. The focus of the criticism was the application language interface, the DML. Discussion of the schema DDL went little beyond an apparent belief that it presented no impediments to distribution. No grounds for such a belief were offered. No discussion at all addressed the adaptability of the DSDL to the description of the distributed environment.

The criticism of the DML was that it was too "low level." This criticism was the strongest result of the discussion of Database Environment and reappears in other areas. Therefore, a summary of the definitions and arguments and the positions submitted by individuals have been collected and appear later in the Database Environment area.

Within the discussion on distributed databases, it was pointed out that minimization, or at least reduction, of channel or network traffic (the amount of data transmitted between nodes of a distributed data network) was a significant goal of any distributed DBMS, since the cost of that traffic may well be the limiting factor in system use. The DML of CODASYL, it was thought, would increase that traffic. The global intention of the application programmer cannot be expressed in the CODASYL DML. This leaves no alternative to the implementation policy of retrieving each record requested by the program so that the selection and aggregation (AVG, MAX, etc.) logic embedded in the program can be

exercised. A higher level DML would allow these low level operations to be distributed. Although this does not decrease the total amount of computation performed, it tends to decrease the volume of data transmitted. The amount of decrease depends on the amount of selection and aggregation implied by the query and by the amount of cross-node comparisons made. (A cross-node comparison references data items stored at distant nodes in the network.) The following positions were submitted on this issue:

> A low-level DML (such as CODASYL DBTG) is inappropriate for a distributed DBMS because the low-level operators, when applied to remote data, result in excessive network traffic. It is necessary to have processes at the remote system to select and aggregate data before transmission. If a high-level DML is used, the system can construct the appropriate processes to be run at the different sites involved.
>
> The alternative of having the programmer code separate low-level processes at each site is not appealing since:
>
> 1. the programmer must be explicitly aware of data location, the programs cannot survive data reorganization;
>
> 2. the coding is much more difficult and must concern itself with intersite communication;
>
> 3. the low-level code must be bound to particular sites. If a site fails, even though redundant data is available elsewhere, the code cannot execute. A high-level global program could automatically use alternate data copies;
>
> 4. with a high-level program, the compiler can optimize the global execution based on database statistics (not possible with a low-level implementation). [Shipman]
>
> The subschema facility, by retaining a close link to DDL schema records and sets, places severe limitations on possible DML. CODASYL 78 represents an improvement over CODASYL 71: for example, set selection by structural constraint permits returning to the user only those records whose contents are of interest; still, placing

this feature in the DDL schema, and thus requiring increased implementation overhead, will limit its usefulness. Most other possibilities for reducing the volume of data transmitted--qualification, statistical reduction, aggregation--are not supported by DDL schema to subschema mappings. Thus these functions will be performed by most language programs, requiring that the necessary data be passed to the user's computer for processing, rather than being reduced before transmission. This results in a substantial increase in the volume of data to be transmitted. [Clemons]

Some argued that global optimization of CODASYL DML programs would be capable of decreasing the volume of network traffic which might otherwise result. However, no one indicated how that optimization might proceed, nor gave any reference to work which had succeeded in finding or attempted to find such an optimization technique.

3.2.2 Special Purpose Machines.

Do associative memories make the definition of access paths like CODASYL sets obsolete? Does the navigational DML make the use of associative memories impossible? Which functions of a DBMS should be allocated to a backend machine? What should be the interface between the backend and the host? How does this fit into the ANSI framework and the CODASYL specifications? This series of questions provoked a discussion on the means by which the specialized devices, e.g., RAP [OZKA], CASSM [COPE], etc., could be incorporated into the ANSI/SPARC framework or into the CODASYL specifications. The term "database machine" encompassed a large variety of specialized non-numeric processors including associative memories, text processors, sorting machines, and so forth. The associative memories under discussion were disk based, "logic per cell" systems which respond to requests for data records given values of constituent data items rather than addresses, and which are capable of limited logic and computation. (The eventual appearance of storage technologies other than disk, specifically bubble memories, in these applications was noted by one of the participants.) Associative memories are examples of backend machines.

The workshop noted that associative memories eliminated the need for complex stored data structures such as inverted files, indices, and so forth. This function of the CODASYL set is not useful in an associative memory environment. As to the adaptability of the CODASYL Schema DDL or data model to associative memories, opinions differed. Some thought

that the relational model was best suited to these devices. Su reviews these arguments:

> Associative memories with context addressing capability are more suitable for the relational model than for the other models because
>
> 1. the relational model represents entity relationships by explicitly storing the primary keys of the related entities (by content) rather than the addresses of the entities, and
>
> 2. the representation of relations as flat tables matches very closely to the architecture of associative memories--thus the effort of mapping the information structure of the data to the structure in the associative memories is minimal.
>
> However, one should not jump to the conclusion that the relational model is superior than other models for the following reasons. First, the capability of associative memories have not yet been fully exploited. With some hardware extension, associative memories can handle more complex data structures such as trees and network efficiently. For example, the CASSM system is built to process hierarchical structures and contains pointer transfer capabilities which are very suitable for data structures using pointers. Second, the hardware should be built and modified to meet the application needs. Hardware consideration should not be a major factor for determining the suitability or the superiority of data models. With the drastic reduction of the hardware cost, one can expect that new hardware (database machines) can be built to support any data model which the users deem suitable for their applications. [Su]

On the DML question, the workshop again criticized the CODASYL specifications as being too low level. Where some had thought that optimization could be of assistance in the distributed environment, the situation was thought to be worse in regard to associative memories. The parallelism of these devices can only be exploited through the use of a higher level DML. When there are "multiple hits" in the associative memory, it was thought by some, the precision of a navigational DML would be required. This was taken to mean that record-by-record or tuple-by-tuple serial operations are a required part of any DML. Others thought that the

ability of a higher level DML to deal with a set rather than an individual returned as the result of a query was a powerful argument for the use of such DMLs. It was noted that associative memories facilitate the implementation and improve the performance of high level DMLs.

The participants generally thought that the more a backend machine did in off-loading work from the host computer, the better. Clearly, there must exist some limit to this off-loading, otherwise the backend simply becomes the host. Most of the comments made by the workshop to this point appear in the position by Su:

> Which DBMS functions should be assigned to the main frame computer and which to the backend processor(s) would depend heavily on such factors as the relative speed, memory size, job load, peripheral device types and speeds, etc. of the processors involved. However, in general, we should expect the main frame to:
>
> 1. carry out the program or query translation task;
>
> 2. support the end-user facilities (ANSI/SPARC);
>
> 3. handle schema mappings, terminal communications, job scheduling and formatting of output data to the users;
>
> 4. control the program execution; and
>
> 5. construct and maintain the data dictionary.
>
> The backend processor(s) should perform the actual data retrieval and manipulation operations, handle the memory management, schedule and control the query or DML execution, enforce the integrity and security rules and implement the data aggregate functions. The data to be passed from the backends to the main processor should be the result of a high-level data operation (a set of records) rather than of a low-level request (one record at a time) to reduce the amount of interference generated to the main processor. [Su]

Goguen added this comment:

> The back-end machine concept places the database management function on a separate processor with exclusive access to the database. The host machine collects data management requests and transmits these across an interface from the host to the back-end. Status and results are accepted from the back-end by the host and sent to the appropriate application programs.
>
> By separating the data management functions and placing them on a specialized processor, the opportunity exists for designing a high level logical interface between the host and back-end processors. With this approach, data can be transferred between different host processors without reformatting and translation, assuming the back-ends are the same. Thus data conversion can be avoided.
>
> A future conceptual schema in the ANSI/X3/SPARC framework could provide the basis for a Standard Data Interchange format which would be the high level logical interface between the host and back-end processors. [Goguen]

The workshop participants appeared to agree that the ANSI/SPARC framework allowed for rapid adoption of specialized processors as they become available.

3.2.3 Higher Level DML.

Having repeatedly criticized the CODASYL DML, the workshop saw the necessity to define more nearly what they meant by "high level DML." A first attempt was that in a high level DML, "you tell the computer what you want;" whereas, in a low level DML, "you tell the computer how to get what you want." This was refined to mean that a low-level DML was procedural or navigational, and a high level DML was non-procedural or non-navigational. However, these terms did not appear to have clearly understood definitions. The following positions were submitted:

> A "navigational" interface to a database management system is an interface which accesses one record at a time according to a specific access path; e.g., "get parent" or "find next in set." A "non-navigational" interface tends to operate on sets of records and to be independent of access path; e.g., "join sales to customers by

matching their account numbers." Like "procedural," the term "navigational" defines a continuum on which I would characterize the hierarchic and network data models as more navigational, and the relational data model as less navigational. [Chamberlin]

If the word "navigation" is to be used to mean any type of program-controlled data traversal, then all DMLs are navigational. They may differ only in the degree of "navigationality." This is because all DMLs are used to traverse and manipulate databases in accordance with the information structures defined in the external schemas. However, some of the structural properties of a database defined in an external schema are syntactic oriented and are inherent from the specific data model used. They are not necessary for defining the semantics of the data. For example, the concepts of database key, area assignment, chaining and sequential access of set members, etc. have more to do with the access path structure of the data than their semantics. If we distinguish that part of the structural properties defined in external schemas which are necessary for describing the semantics of databases from that part which are not, we can then define a navigational DML as a language which allows the user to guide the data traversal through the excess structural properties and a non-navigational DML as one which allows the user to retrieve and manipulate data based only on the semantic properties of the database. [Su]

It is difficult to establish that there exists any language which is not navigational at all; probably the fact is that some languages are more navigational than others. A potential measure of the degree to which a language is navigational is the following: "Every language statement implicitly contains in its semantics a data structure which it addresses; the closer this data structure is to the actual schema structure, the higher is the degree to which the language is navigational." [Lefkovits]

Rather than directly address level of DML, which seems to be an ill-defined and elusive concept, I address related subtopics: iteration and navigation. Iteration in DML is the ability to write a single DML statement that retrieves and

processes all desired record occurrences. For example, when retrieving the arithmetic average of salary of all employees of a department, DML iteration would permit this to be done without explicit host language looping and arithmetic statements. Although such facilities are not provided in the current CODASYL DML, they could be added with few changes to DDL schema and subschema facilities. I define database navigation to be the process by which the user issues DML statements to make associations between or among desired records. This definition is independent of data model and language used--it applies equally to navigation based on set membership, algebraic joins or SEQUEL mappings. Although all implementations require some navigation, user navigation is bad, because it is time-consuming, difficult, and likely to result in errors. The alternative is replacing DML association statements with DDL statements, that is, making data associations and accesses part of the external schema definition. This is readily done as an extension to the relational model, as shown by System R and, to a greater extent, my own research. It has not been done by CODASYL: no such facilities are included in the subschema facility. Moreover, extensions needed to accomplish this may be quite complex due to the basic set architecture chosen; the SELECTION clause of CODASYL 78 DDL demonstrates this. [Clemons]

Zaniolo provided an overview that linked several of the topics in order to reveal additional insights:

> The ANSI/X3/SPARC architecture is conducive to data independence, powerful end user facilities and high level DMLs suited for structured programming. Its framework will enable a DBMS to respond well to environment changes including distributed data and database machines. The following features are most important in realizing the previous benefits: three levels of schemas, multi-model external schemas, external schemas derived from other external schemas, dictionaries. While CODASYL 78 uses three levels of schemas it does not provide the remaining facilities which, I believe, are necessary to realize the benefits previously described. I see no conceptual problem in adding a dictionary facility specification to the CODASYL 78 documents. However, the addition of more powerful

external schema facilities could produce strong repercussions in the schema DDL required to support such mapping flexibility.

The objective of this statement is not to pass judgment on the relative merits of CODASYL 78 and ANSI/X3/SPARC, but to suggest that a distinction between the two would be useful in clarifying the discussion and the conclusions of this workshop.

HIGH LEVEL DMLs: These languages are very useful since they facilitate application programming and promote practical data independence. The discussion at the workshop focused on two aspects of high level DMLs:

i. Aggregate Operation Aspect. This refers to the capability of manipulating and retrieving a whole set of records, selected on the basis of their content, in a single DML statement.

ii. Non-Navigational Aspect. (The definition of navigational languages is given below).

Current query languages (e.g. the QLP language for DMS/1100 of Sperry Univac) support aggregate operations against network schemas. However, non-navigational DMLs will probably require external schemas different from the network subschemas of CODASYL 78.

NAVIGATIONAL LANGUAGES: These are languages based upon the availability of access paths in the application schema. Thus the user identifies paths and records of interest and specifies his query as steps thru them. While access paths can be defined as purely logical constructs, they have traditionally been regarded as expression of underlying physical structures, thus impairing data independence. Moreover a query on a relationship not directly supported by an access path is expressed differently and less conveniently than a query on a supported relationship. [Zaniolo]

The participants agreed, as implied by Chamberlin's statement, that a high level DML takes as its operands and returns as its results aggregates of records, tuples, entities, relationships, or whatever, rather than individuals.

Although no consensus was reached on how non-procedurality is recognized, however, it was agreed that procedurality and navigation were matters of degree. Some participants felt that insofar as the means of establishing a relationship (CODASYL sets or relational joins) between objects (records or tuples) appeared in the DML, the DML was navigational. This led to the position, not submitted as a statement, that the hierarchical model allowed for the least navigational DML. The workshop did not agree as to whether the choice of data model dictated the degree of navigation or procedurality on an associated DML. Some participants thought that a non-procedural, non-navigational DML for use with a CODASYL organized database was conceivable; others did not. Dave Shipman took what might be called a higher level view of higher level DMLs:

> What is a High-Level Language? A higher-level language is one in which the contents of the language more closely resemble the terms in which the user thinks of his problem. This is accomplished by embedding the semantics of the problem domain within the programming language itself, in terms of DBMS languages. This means that programs in a high-level DML express the intent of the query or update, rather than the sequence of low-level operations needed to carry it out. A high-level DML pre-supposes a DDL facility which allows the semantics of the application to be reflected in data descriptions rather than in each DML program which references those descriptions. A high-level DDL in turn incorporates constructs natural to modelling the real world enterprise. High level languages are easier to write, debug, document, maintain, and administer. In addition, because a high-level language is less bound to any particular hardware or software implementation, new technologies can be absorbed without significant impact on existing application programs. [Shipman]

3.2.4 Advanced Programming.

Do either the ANSI/SPARC framework or the CODASYL specification inhibit database programmers in the use of the methodology known as structured or modular programming? Can the results of research into abstract data types be applied to database problems?

These questions attempted to elicit the participants' opinions of recent advances in software engineering that have some impact on database engineering. Discussion of the topic was brief. It was felt that the ANSI/SPARC framework "blends nicely into the ideas of abstract data type" but there were no further details forthcoming. Concerning the CODASYL specifications, it was remarked that their complexity did not lead to elegant programming style.

A number of the participants felt the need for database operations defined on the semantics of the data and that the work in abstract data types could be used as an example. Consider the contrast between the commands STORE EMPLOYEE RECORD and HIRE EMPLOYEE. The first is expressed in terms of the data requirements of the DBMS. The second is expressed in the semantics of the enterprise. The second command might include side effects such as establishing relationships (Department-Employee), updating counts (Number of Employees in Department), and so forth.

3.3 END USER FACILITIES

The CODASYL specifications to date have not addressed the end user's needs. The ANSI/SPARC study group identified four interfaces for the end user, but did not specify them in detail.

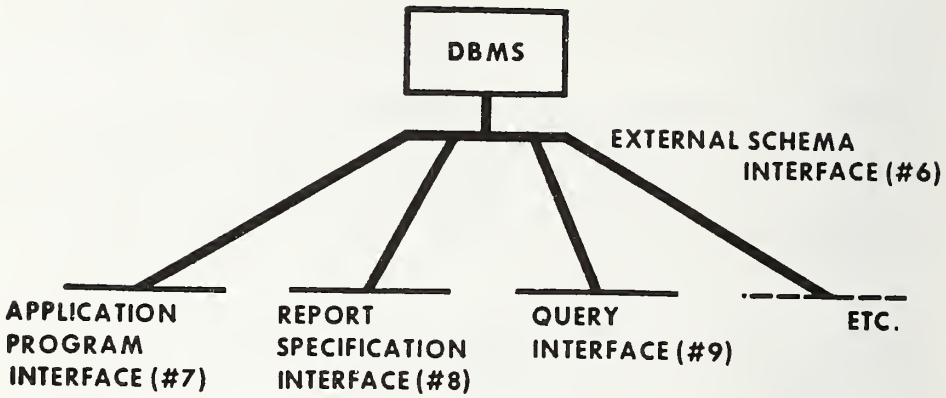
3.3.1 Is the ANSI user analysis correct?.

The ANSI/SPARC report recognizes four separate interfaces, (numbered 8 through 11), for the end user. They are the Inquiry Processor (small output volume, interactive); Report Writer (large output volume, batch); Update Processor; and Parametric interface.

The majority of the participants considered this analysis to be incorrect. Don Chamberlin points out in the following statement that the application programmer interface, interface 7, is presented as being parallel to the end user interfaces, which he considers an error:

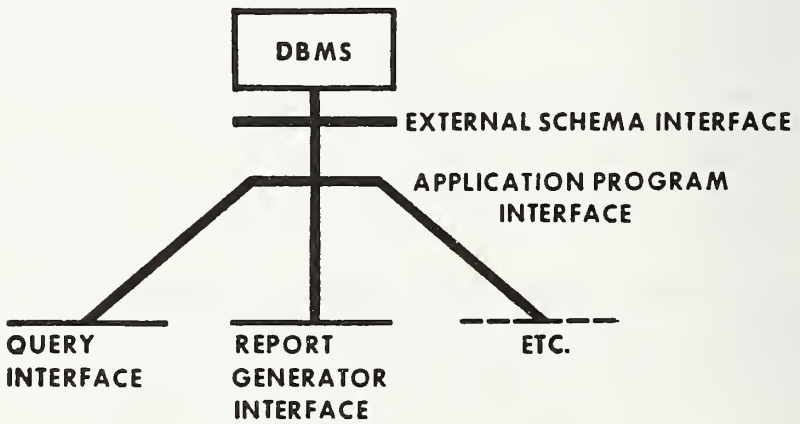
> Role of Application Programming Interface in a DBMS
>
> The ANSI/SPARC framework for a DBMS [ANS77] treats the application programming interface (API) as independent of the query, report generator, and several other interfaces (Figure 5).

Figure 5
Application Programmer Interface



An alternative which has several advantages is to make the API sufficiently rich to enable programs to be written in support of query, report generation, etc. (Figure 6).

Figure 6
Enriched Application Programmer Interface



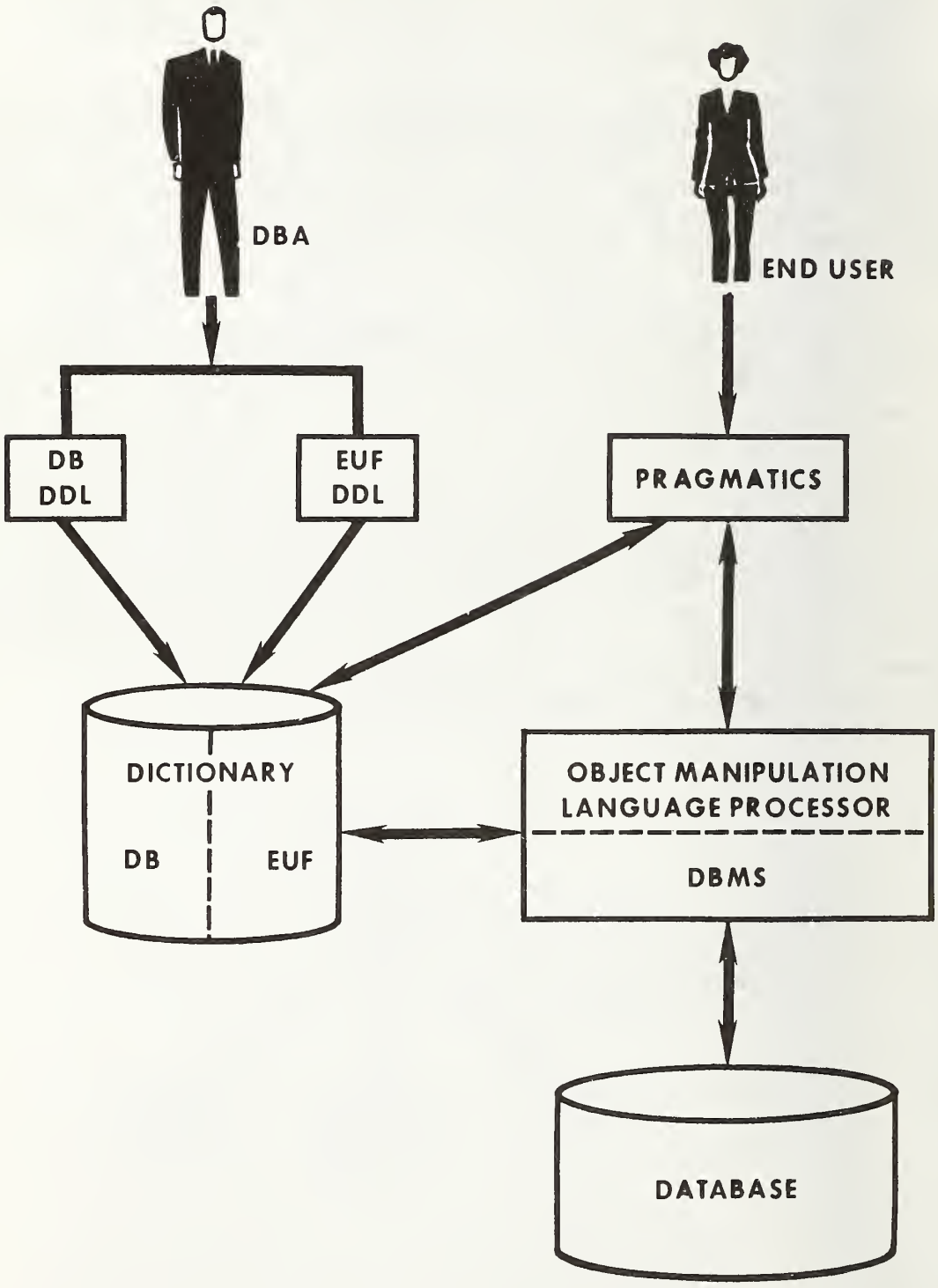
Such an enriched API would need facilities for

> submitting ad-hoc queries and updates for execution (e.g., the EXECUTE statement of SQL [CHAM]). A system structured according to Figure 6 permits a variety of query and report generator interfaces to be developed and supported by programs, to meet a variety of end user requirements. [Chamberlin]

Other participants expressed the view that the four separate interfaces of the ANSI/SPARC diagram represented a data processing solution, in sharp contrast to a user oriented solution. In other words, these four operations appear separate and distinguishable to a computer-knowledgeable data processing professional, but not to an end user.

Henry Lefkovits drew a diagram (Figure 7) for the design of an End User Facility or Facilities. The diagram results from the work of the End User Committee of CODASYL of which Dr. Lefkovits is the chairman. Although that committee has yet to publish a Journal of Development, their interim report [EUFTG] indicated that they were developing specifications for a facility based on the "forms approach." Explanation of this approach, in which the database is viewed as a "virtual file cabinet" within which the data is organized into forms resembling the familiar paper forms on which business has so long relied, led the workshop to renewed criticism of parts of the CODASYL JOD specifications. This discussion is given in the answer to the next question presented to the workshop.

Figure 7
Design of an End User Facility



3.3.2 Does subject architecture impede EUF?.

Many participants agreed that the subject architecture (the three schema framework of ANSI/SPARC implemented in the CODASYL 78 language) presented impediments to the construction or use of end user facilities.

The need recognized by the End User Committee to define an end user view of the database reinforced the beliefs held by many of the workshop participants that the CODASYL COBOL Subschema DDL was not sufficient to serve the needs of a complete external schema defining language. It was stated by one of the participants that the "subschema facility ... is fundamentally misdesigned [and] entirely inappropriate" for use as an external schema facility. Don Chamberlin summarized the views of many in the following statement:

> 1. CODASYL subschema DDL is not a very rich facility for defining views of stored data. Essentially, the defined views must be simple subsets of the schema.
>
> 2. Because CODASYL DDL lacks the "closure" property, it does not permit "cascading" of views (views defined on top of views). [Chamberlin]

The ability to cascade was seen by some as more a matter of convenience than necessity: the final view or subschema or external schema could be defined directly from the schema. However, it was observed that forming views on views increased security and access control, and provided a more flexible division of labor between enterprise and application administrators, as those roles are defined by ANSI/SPARC.

Some debate addressed whether a sufficiently powerful external schema facility to support most users' needs could be designed to interface with the CODASYL schema DDL, with the majority apparently believing that it could.

It was remarked that neither the ANSI/SPARC nor CODASYL documents made any mention of text processing nor of message management (communicating among end users) and that these were important features of an end user facility.

3.3.3 Is EUF different from programming interface?

The participants were, for the most part, convinced that end user facilities did not present any hinderences to database evolution beyond those presented by application programs. It was pointed out, however, that the end user

environment is likely to be less controlled than the data processing environment. Ad hoc database use may create more unpleasant "surprises" than might occur in the absence of end user interfaces. Good intra-enterprise communication and special attention to the preservation of end user views are two ways to deal with surprises.

3.4 DATA DICTIONARIES

In the center of the block diagram published by the ANSI/SPARC study group there is an unlabeled triangle from which emanate a large number of the interfaces recognized and described by that group. This triangle is the data dictionary, and its role is apparently crucial to the ANSI/SPARC framework. The study group report gives no details of a dictionary system beyond the image of a repository for whatever control information is needed by the DBMS.

Recent work by the British Computer Society Data Dictionary Systems Working Party [BCS] presents an elaborate and powerful facility which they called "the data processing department's own database." Members of the workshop were quick to point out that the data dictionary/directory (DD/D) should serve not only the data processing department but the larger user community as well. They also criticized the ANSI/SPARC report for failing to see that the dictionary interfaced to the human users of the database. Most of the workshop participants agreed that a data dictionary system had two distinct aspects: one oriented towards use by the DBMS; and one oriented towards use by human beings. The nature of the data stored in a DD/D and the services performed by it reflect this dual role.

3.4.1 What is the data stored in a DD/D?.

Everything about the data stored or storable in the database of interest to the enterprise. Everything except the data values and programs themselves. As concrete examples:

- The three schemas and their associated mappings in source and object form.
- Security, access control, authorization mechanisms.
- Usage information. This was of two types: static information as to which data items are used by which programs; dynamic information concerning frequency and volume of use.

- Validation: range checks, permissible values, etc.

- Narrative text describing data items, aggregates, etc.

- An example of human oriented data not of interest to the DBMS.

An argument was made that the system being described by this list was broad, too all encompassing. It seemed the system would "do everything for everybody" although none had described how it would manage. Of the DBMS-oriented data, it was argued that since any functioning DBMS kept that information, calling it a data dictionary served no new purpose. The reply was that the processing of this information for results not derivable from any single piece of information is a new service provided by DD/D systems.

3.4.2 Languages used with a DD/D system.

Three distinct languages were proposed. They were a customizing language, a maintenance language, and a reporting language. This analysis was accepted by the workshop with the following criticisms and clarifications:

The customizing language need be no more complex than the options selected at system generation. The only customizing decisions to be made are the choices of optional system services.

Reporting from the dictionary is conceptually no different than reporting from the database. The same facility can be used to do both. This prompted a discussion on the merits of organizing the dictionary as a database under a DBMS. On the one hand, it was felt that the dictionary is information, and a database is a box for storing information. It seems reasonable to store the information about the database in the box. On the other hand, some participants perceived a marginal increase in security if the dictionary were not stored under the DBMS. The workshop was reminded that some functions of the dictionary were of use in the absence of a DBMS. Most workshop members thought the use of the DBMS to store the dictionary a sensible decision.

Maintenance of the dictionary, it was agreed, differed substantially from maintenance of the database. Two causes of this difference appeared:

- Some of the maintenance is done by the system itself. Examples of this include dynamic usage information as previously mentioned. Other processes of the system may maintain some information in the

dictionary. The workshop did not discuss the importance of automatic versus manual data collection in the DD/D.

- The effects of changes in the dictionary are potentially much greater than the effects of changes in the database. There is a considerable difference between adding a domain to the description of a relation, and adding a tuple to the extension of a relation. It is correct, therefore, to restrict maintenance of the dictionary to be a single, controllable facility.

3.4.3 Services performed by the DD/D.

The participants suggested several basic DD/D services:

- Resolving name conflicts. These are of two kinds: two (or more) names for the same thing; and the same name for two (or more) different things. It is the effort of collecting this information for the dictionary which recognizes these conflicts.

- Generate data descriptions: COBOL FD's, PL/1 data structures, etc. All, or nearly all, dictionary products on the market perform this service.

- Query cost estimation and access path selection. The data needed for this service includes: Availability and selectivity of indices (the selectivity of an index is the number of unique key values divided by the number of objects indexed); the cardinality of the stored relations; and the average chain length. The workshop did not define the conditions under which the cost of maintaining this data outweighs its usefulness.

- Audit trails. Not to be confused with the update/change logs kept for database backup and recovery. Two types of audit trails were discussed:
  1. a list of which programs are capable of modifying which data items. This can be very useful for tracking down the source of improper database updates; and
  2. a list of the changes made to database programs in production and development systems.

- Impact of proposed changes. A list cross referencing the data items (attributes, entities, and relationships) and the programs which access them. This can help the database administrator avoid mistakes.

- Data Dictionaries.
  Use of a data dictionary can be a significant factor in data conversion. The data dictionary provides a catalog of the data, both on the item level and in terms of associations amongst the data. Assuming that a complete data description is included in the data dictionary, the data conversion system can use the information for generating the appropriate source and target data descriptions required for the conversion. A major problem in conversions to date has been the unavailability of this level of description in usable form. The definition has been implicit in the logic of the application programs or the data structures, requiring time consuming analysis to generate the data description. The ANSI/X3/SPARC architecture should include provision for a data dictionary facility. [Goguen]

- DBMS were introduced to administer and control data which had until then been allowed to accumulate in unmanaged file systems. DBMS succeeded in being data management systems in only a narrow sense. They manage to control and integrate program access of the data. Data dictionary systems are now being introduced to administer and control database management systems.

In summary, the areas in which a dictionary system serves a function were seen to be fourfold:

- Operational: the support of production systems through security, validation, and audit trails.

- System development: the support of application programs under development.

- System installation: can be used to hold the "before and after" descriptions of databases being translated/converted.

- Conceptual Schema or Database Design: useful in all stages.

3.5 TRANSLATION/CONVERSION

The established user of a DBMS will have invested a considerable sum in the data and the programs which access and maintain it. Inevitably, changes in all areas affecting the DBMS will threaten that investment. These changes come from the enterprise itself, from advancement in technology, or from growth. The object is to minimize the dollar cost of these changes.

Two approaches can be distinguished. These are tentatively called the physical and the logical approach. ("Physical" is not to be taken as a synonym for "brute-force" nor is "logical" to be confused with "correct.") The physical approach is the actual, physical translation of the data from the old format to the new, and the actual rewriting of the programs to the new environment. The logical solution is the provision of new schema to schema mappings which preserve unmodified the old view of the data. It may be said that the cost of the physical solution is paid all at once whereas the cost of the logical solution is paid over time.

3.5.1 Database Translation.

Does the ANSI/SPARC framework or the CODASYL specifications inhibit the task of data translation? Specifically, research into data translation has shown the need for Standard Data Interchange Format (SDIF) [NBS]. How does this fit into ANSI/SPARC or CODASYL?

The ANSI/SPARC framework was thought an excellent vehicle for data translation. The SDIF was identified with the conceptual schema. This began a discussion of the requirements of an SDIF. Like the conceptual schema, the SDIF should describe the semantics of the data explicitly. A set of operations on the semantic constructs of the SDIF should be defined. The ANSI/SPARC report was criticized for not providing for these operations. The transformations which bring the source database expressed in the SDIF into the target database is described by an expression in these operators.

There was some argument that this approach was a case of "overkill." The complete semantics of the data are not required in the specification of those formal transformations needed to modify data structure. A semantically complete conceptual schema facility might facilitate data translation, but it is not necessary.

The CODASYL specifications were felt by most workshop participants to be inadequate to play the role of an SDIF. On the one hand, too much physical detail is present. On the other hand, not enough semantic detail is present. A number of alternative means of representing semantics were discussed, of which the Entity-Relationship model of Peter S. Chen was considered by some to be appropriate.

3.5.2 Program Conversion.

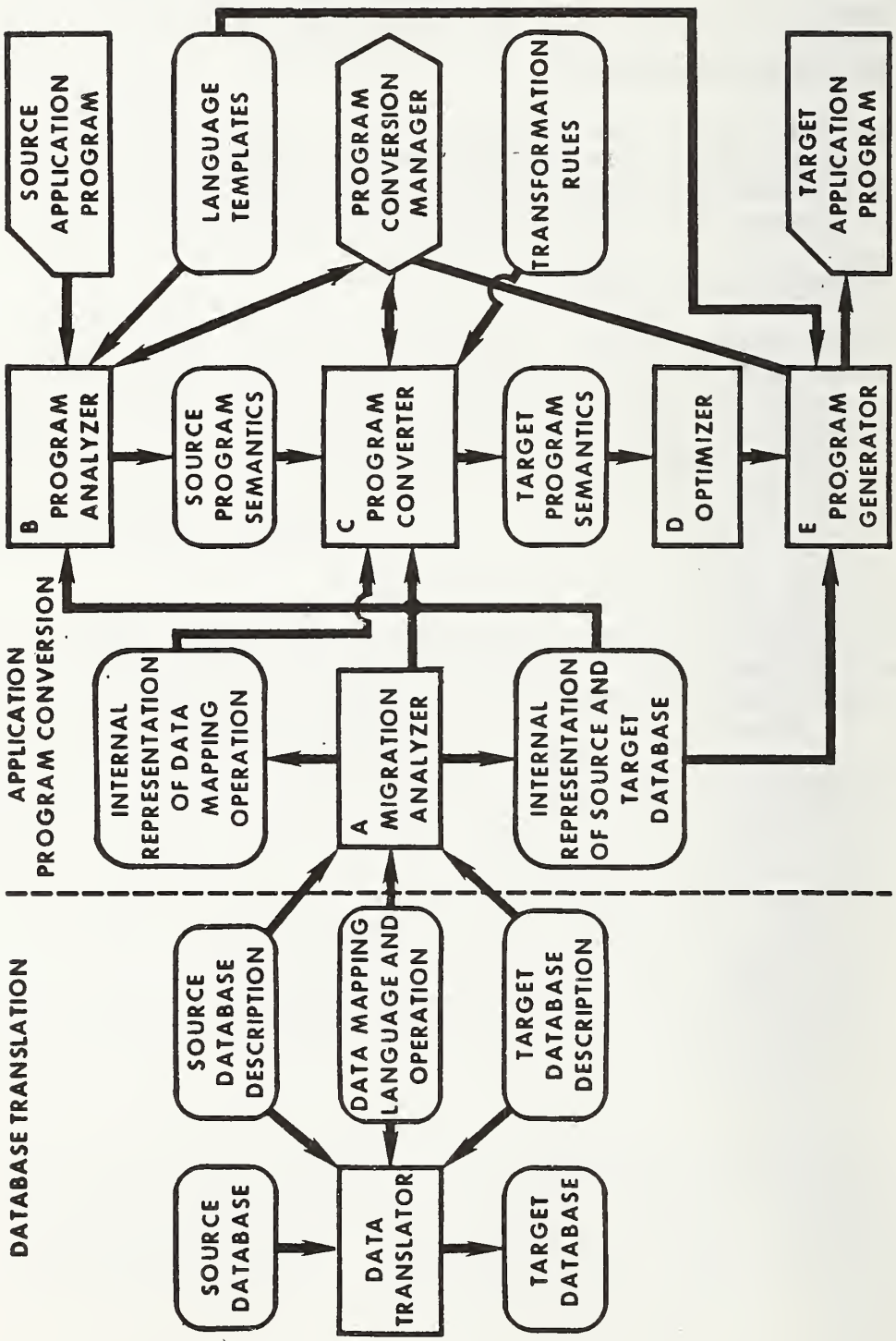
The need for a description of the intention of the programs under conversion, expressed at a high level, was noted. This brought renewed criticism of the CODASYL DML. Su, based on recent work [SULO], drew the diagram in Figure 8.

3.5.3 Dynamic Conversion.

The workshop participants had already criticized the CODASYL subschema as inadequate as an external mapping facility. They did not comment on the feasibility of internal mappings which would allow a DBA to choose not to undergo physical data translation. They felt that internal mappings should be kept simple since, unlike external mappings, they must be invertible. Where a given external view may not allow update, changes to the conceptual database must always be reflected in the physical database.

An approach called "dynamic restructuring" [GERR] which combines aspects of the physical and logical approaches was mentioned. Stored records are marked with a generation number. Records which are retrieved bearing an outdated generation number are converted to the latest format before being stored.

Figure 8
Conversion Processes



3.6 CONCLUSIONS

The group reached the following general conclusions:

1. The ANSI/SPARC architecture was well suited for providing data independence with respect to anticipated technological development in computer hardware and software.

2. Development of end user facility does not hinder data independence.

3. A data dictionary properly used provides a measure of data independence with any DBMS architecture.

APPENDIX A

REFERENCES

[ANS75], "The ANSI/X3/SPARC DBMS Framework, Interim Report of the Study Group on Database Management Systems," FDT, Vol 7, No. 2, 1975.

[ANS77], "The ANSI/X3/SPARC DBMS Framework, Report of the Study Group on Database Management Systems," Tsichritzis & Klug, ed., AFIPS Press, 1977.

[BCS], "The British Computer Society Data Dictionary Systems Working Party Report," SIGMOD RECORD 9:4, December 1977.

[CHAM], Chamberlin, D.D., Astrahan, M.M., Eswaran, K.P., Griffiths, P.P., Lorie, R.A., Mehl, J.W., Reisner, P., Wade, B.W., "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control," IBM J. of R & D, 20, No. 6, November, 1976.

[CHEN], Chen, P. P-S, "The Entity-Relationship Model -- Towards a Unified View of Data," TODS 1,1 (March 1976), pp 9-36.

[COPE], Copelan, G.P., Lipovski, G.J., Su, S.Y.W, "The Architecture CASSM: A cellular system for non-numerical processing," Proceedings of the First Annual Symposium on Computer Architecture, December 1973, pp 121-128.

[DATE], Date, C.J., "An Introduction to Database Systems, Second Edition," Addison-Wesley, Reading, MA, 1977.

[EUFTG], End User Facility Task Group of CODASYL Systems Committee, "Progress Report," FDT (Bulletin of the ACM Special Interest Group on Management of Data), 8, No. 1, 1976.

[GERR], Gerritsen, R, Morgan, H.L., "Dynamic Restructuring of Databases With Generation Data Structures," ACM 76, pp 281-286.

[GUTT], Guttag, J, "The Specification and Application to Programming of Abstract Data Types," Ph.D. Thesis, Department of Computer Science, University of Toronto, September 1975.

[JARD], Jardine, D.A., "Principles of Data Independence," Proceedings of the SHARE Working Conference on Data Base Management Systems, Montreal, July 1973.

[LISK], Liskov, B, Snyder, A., Atkinson, R., Schaffert, C., "Abstraction Mechanisms in CLU," CACM 20:, August 1977, pp 564-576.

[NBS], Fry, J.P., et al., "An Assessment of the Technology for Data- and Program-Related Conversion," AFIPS NCC 78, Volume 47, pp 379-387.

[OZKA], Ozkarahan, E. A., Shuster, S.A., Smith, K. C., "RAP - An Associative Processor for Database Management," AFIPS NCC 75, Volume 44, pp 379-387.

[SENK], Senko, M.E., "Specification of Stored Data Structures and Desired Output Results in DIAM II with FORAL," Proceedings [First] International Conference on Very Large Data Bases, September, 1975, (available from ACM), pp 557-571.

[SOCK], Sockut, G.H., Goldberg, R.P., "Data Base Reorganization - Principles and Practice," NBS Special Publication 500-47, April 1979, Computing Surveys, Vol. 11 No. 20, December, 1979.

[STON], Stonebraker, M. and Held, G., "Networks, Hierarchies and Relations in Data Base Management Systems," University of California Electronic Research Laboratory Memorandum No. ERL-M504, March 1975.

[SULO], Su, S.Y.W., Lo, D.H., Lam, H., "Application Program Conversion Due to Semantic Change," Report 7879-2, Computer and Information Science Department, University of Florida, March 1978.

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)

114A (REV. 2-8C)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET (See instructions)	1. PUBLICATION OR REPORT NO. NBS SP 500-76	2. Performing Organ. Report No.	3. Publication Date April 1981

TITLE AND SUBTITLE

Database Architectures--A Feasibility Workshop Report

AUTHOR(S)

John L. Berg, Marc Graham, and Kevin Whitney (Editors)

PERFORMING ORGANIZATION (If joint or other than NBS, see instructions)

NATIONAL BUREAU OF STANDARDS
DEPARTMENT OF COMMERCE
WASHINGTON, D.C. 20234

7. Contract/Grant No.

8. Type of Report & Period Covered

Final

SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (Street, City, State, ZIP)

Same as item 6.

SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 81-600004

[ ] Document describes a computer program; SF-185, FIPS Software Summary, is attached.

ABSTRACT (A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here)

To help the decision maker evaluate the potential benefits and pitfalls in moving forward with database technology, the National Bureau of Standards organized two workshops whose results are presented in this report. The workshops, held in August 1978, explored the progress plan and potential pitfalls involved in specifying, designing, and implementing systems based on the ANSI/X3/SPARC framework and the CODASYL JOD languages specification. Workshop 1 investigated the general topic of data independence, and Workshop 2 examined supporting topics such as query languages, data dictionaries, and database conversion.

KEY WORDS (Six to twelve entries; alphabetical order; capitalize only proper names; and separate key words by semicolons)

Conversion; Database; Data-description; Data-dictionary; Data-directory; Data-manipulation; DBMS; Language; Query; Standards.

AVAILABILITY

[X] Unlimited

[ ] For Official Distribution. Do Not Release to NTIS

[X] Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402.

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

14. NO. OF PRINTED PAGES

64

15. Price

$4.00

USCOMM-DC 6043-P80

NBS TECHNICAL PUBLICATIONS

PERIODICALS

)URNAL OF RESEARCH—The Journal of Research of the ıtional Bureau of Standards reports NBS research and develop-nt in those disciplines of the physical and engineering sciences in ich the Bureau is active. These include physics, chemistry, gineering, mathematics, and computer sciences. Papers cover a)ad range of subjects, with major emphasis on measurement thodology and the basic technology underlying standardization. so included from time to time are survey articles on topics sely related to the Bureau's technical and scientific programs. a special service to subscribers each issue contains complete ations to all recent Bureau publications in both NBS and non-S media. Issued six times a year. Annual subscription: domestic ; foreign $16.25. Single copy, $3 domestic; $3.75 foreign.

)TE: The Journal was formerly published in two sections: Sec-n A "Physics and Chemistry" and Section B "Mathematical ences."

MENSIONS/NBS—This monthly magazine is published to in-m scientists, engineers, business and industry leaders, teachers, dents, and consumers of the latest advances in science and nology, with primary emphasis on work at NBS. The magazine hlights and reviews such issues as energy research, fire protec-n, building technology, metric conversion, pollution abatement, lth and safety, and consumer product performance. In addi-n, it reports the results of Bureau programs in measurement ndards and techniques, properties of matter and materials, ineering standards and services, instrumentation, and)matic data processing. Annual subscription: domestic $11; eign $13.75.

NONPERIODICALS

nographs—Major contributions to the technical literature on ous subjects related to the Bureau's scientific and technical ac-ies.

ndbooks—Recommended codes of engineering and industrial ctice (including safety codes) developed in cooperation with in-sted industries, professional organizations, and regulatory ies.

cial Publications--Include proceedings of conferences spon-d by NBS, NBS annual reports, and other special publications ropriate to this grouping such as wall charts, pocket cards, and iographies.

lied Mathematics Series—Mathematical tables, manuals, and lies of special interest to physicists, engineers, chemists, logists, mathematicians, computer programmers, and others aged in scientific and technical work.

ional Standard Reference Data Series—Provides quantitative on the physical and chemical properties of materials, com-d from the world's literature and critically evaluated. eloped under a worldwide program coordinated by NBS under authority of the National Standard Data Act (Public Law 96).

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 amende Public Law 89-306 (79 Stat. 1127), and as implemented by F ecutive 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.

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